AUDIOVISUAL PHYSICS-BASED SIMULATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Timothy Richard Langlois

August 2016

AUDIOVISUAL PHYSICS-BASED SIMULATION

Timothy Richard Langlois, Ph.D.

Cornell University 2016

Physical simulation has been used for decades to gain insight into physical phenomena, evaluate stability of designs, and in computer graphics to reduce artist effort. With increasing computational power and algorithmic improvements, we have not only been able to simulate phenomena in higher resolution and more complex domains than ever before, but to use simulation as a general problem solving tool. This thesis explores the use of audiovisual physical simulation techniques in several directions: creating better design tools, making physical simulation more accessible, and using simulation to gain insight into mathematical models and physical processes.

First, we explore the inverse problem of automatically generating an animation that is synchronized to a recorded sound. Our main insight consists of sampling a large number of rigid body simulations, and treating them as a *contact-event graph*. Paths in this graph correspond to different animations. We show how to efficiently search this graph to find plausible animations synchronized with an input sound. This provides a new way to design and control animations, while taking advantage of the fidelity of real sounds which can be difficult to capture with physically based sound synthesis.

Second, we examine one way to make physical simulation easier to use. Modal sound synthesis for rigid objects has seen widespread success due to the ability to precompute vibration modes and radiation fields. However, its speed

comes at a cost: namely the large amount of memory required to store the mode shapes. We demonstrate how to significantly compress the modes by fitting a moving least-squares approximation to them. The approximation error is set to take advantage of human perception, and we also exploit object symmetry to achieve higher compression.

Finally, we explore using physical simulation to synthesize water sounds, which are mainly caused by bubble volume vibrations. We introduce a method to calculate the frequency of bubble vibrations, showing how to accurately take the bubble's size, shape, and position into account. In addition to generating compelling animations, this helps evaluate the quality of current bubble forcing and damping models, and will hopefully provide a reference to judge future approximations by.

BIOGRAPHICAL SKETCH

Tim Langlois was born and raised in Tyngsboro, Massachusetts. His first interest in technology came from playing *Where in the World is Carmen Sandiego* on 5¼ inch floppies (especially with the turbo button on his family's Packard Bell – 25 MHz!). He received a B.S. from UMass Amherst in 2009, where he worked on problems in computational biology and built an AI/vision system to assist billiards players. For two summers during his undergraduate career, he interned at DEKA Research & Development in Manchester, NH, where he worked on a robotic prosthetic arm project intended to help wounded veterans. From 2009 to 2011, Tim was a software engineer in the weather sensing group at MIT Lincoln Laboratory, where he worked on a short term, high-resolution weather forecast system that the FAA uses for planning air traffic. Since 2011, Tim has been studying for his doctorate degree in Computer Science at Cornell University.

This thesis is dedicated to my wife Qiao Zhou. Without your endless love, support, and sacrifice, this would not have been possible.

周桥, 我爱你

ACKNOWLEDGEMENTS

First and foremost, a sincere thank you to my advisor Doug James, who has been a neverending source of wisdom, encouragement, inspiration, and wonderful deadpan humor. Doug has helped me to grow not only as a researcher, but as a person. I would also like to thank my other collaborators Changxi Zheng, David Levin, Steven An, Kelvin Jin, and Ariel Shamir – and my other Ph.D. committee members Dexter Kozen and John Guckenheimer. The Cornell Computer Graphics Lab has been an amazing place to work and learn, mainly due to all the great people I have interacted with and learned from: Steve Marschner, Kavita Bala, Noah Snavely, Sean Bell, Pramook Khungurn, Jeff Chadwick, Eston Schweickart, Jui-hsien Wang, Kevin Matzen, Daniel Hauagge, Kyle Wilson, and Scott Wehrwein.

I would also like to thank Ramgopal Mettu, Sai Ravela, William Dupree, and Marilyn Wolfson. The time I spent at UMass Amherst and MIT Lincoln Laboratory was invaluable, and ultimately helped me decide to persue graduate school. The most important step on any journey is the first.

Finally, I am indebted to my family. My parents, siblings, grandparents, aunts, uncles, cousins, and everyone else who I'm not sure how I'm related to but still keep showing up at Christmas created a wonderful and supportive environment to grow up in. I'm sorry I had to be away from you all for so long. And to my wife Qiao Zhou: you are the strongest, most beautiful person I know, and have put up with more than anyone should have to. I'm not sure what I did to deserve you, but I'm glad I did it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Physical simulation has long been used in various fields to understand physical phenomena, to aid in designing objects to withstand real world conditions, and as a creative content generation tool. For decades, the field of computer graphics has developed and utilized physical simulation as a means to add detail to animations while reducing artist effort. The earliest interests focused on the simulation of visual phenomena. This continues to be the main focus of computer graphics simulation, and has seen success in various areas such as rigid bodies [10, 11, 64, 19, 55, 143], fluids [147, 59, 60], deformable objects [156, 155, 14, 133], smoke [57], fracture [113, 171], rods and threads [21, 20], cloth [12, 80], muscle and tendons [134, 150], ice [82], lightning [83], and snow [148]. As algorithms have improved, and computational power has increased, simulation has been utilized in novel ways, including interactive simulation editing [15], learning character controllers such as dressing [39], bicycle stunts [152], and swimming [153]. Simulation has also been used in multiple ways to improve 3D printing, such as designing objects with specific audio properties [22] and hydrographic printing [169].

To make simulation a useful tool for content generation, users often require artistic control of animations. A variety of techniques for control of rigid body animations have been proposed, including spacetime optimization [167, 40], interactive navigation [122], motion sketching [123], and various sampling methods [154, 36, 161, 160]. Character animation is commonly controlled with motion graph techniques [86, 91, 7, 81, 90]. Control methods for deformable objects have included motion graphs [77], spacetime constraints [67] and optimization [13], and rest pose adaptation [43]. For fluids and smoke, methods include

keyframing with multiple shooting [157], the adjoint method [104], and guide shapes [111].

Sound is also an important aspect of virtual experiences. The default method for adding sound to film and animation has been foley. Named after Jack Foley, who began adding sound to silent movies at Universal Studios in the 1920s, foley consists of recording sounds and manually lining them up to events in movies or animation. Recently, there has been work on simulating the sound that objects in animations make, in order to automatically generate realistic, synchronized sounds and reduce foley effort. Sound is pressure waves that travel through the air (or other mediums). The simulation of sound can be roughly split into two main areas: synthesis, which describes how objects generate pressure waves; and propagation, which describes how pressure waves travel through the environment.

The main way solid objects generate sound is through surface vibrations. Early work in computer graphics explicitly resolved these vibrations through finite element simulations integrated at audio rates [114]. The special case of rigid objects can be solved efficiently through a precomputed modal decomposition, and has attracted a moderate amount of interest [115, 75, 25, 172, 165]. Other types of objects and phenomena such as thin shells [32], fire [33], cloth [5], fracture [171], acceleration noise [35, 34], and water [170, 110] have required special treatment due to the various and complex ways in which they generate sound.

In addition to how objects generate sound, compelling audio simulation requires modeling the propagation of sound waves. Early work in computer graphics utilized geometric acoustics methods such as beam tracing [61] possibly including edge diffraction [158]. These models work well in the high frequency

regime, but at mid and low frequencies diffraction effects become important. More recent work has focused on simulating wave propagation to capture these effects [128, 130]. For rigid objects in free space, propagation is usually computed by solving the Helmholtz equation for each mode shape. Solutions are precomputed for each mode using a boundary element method, capturing complex boundary effects and allowing efficient runtime evaluation [75].

In this thesis we explore several problems at the intersection of audio and simulation, and explore how audiovisual simulation can be used as a tool to do more than simply simulate physical phenomena.

Chapter 2 explores the inverse problem of synthesizing an animation to match a recorded sound. This allows us to leverage the realism of recorded sounds, which can be difficult to reproduce with physically based sound synthesis. It also provides a method of motion control based solely on sound.

Chapter 3 solves one of the longstanding problems with modal sound models: the need to store the entire mode matrix in memory at runtime. The memory requirements limit the size (larger objects have more modes) and number of objects that can be simulated. By fitting moving least squares approximations to mode shapes, and exploiting symmetry and human perception, we are able to achieve up to 1000x compression ratios without any audible degradation.

Chapter 4 investigates the simulation of water sounds, which are generated primarily by bubble volume vibrations. While previous methods have been proposed in computer graphics, they have relied on ad-hoc methods for frequency estimation and idealized bubble models. We demonstrate the first method to accurately account for the complete frequency effects of bubble size, shape, and position. This allows us to generate more realistic fluid sounds, with more of a

basis in physics than previous methods.

CHAPTER 2

INVERSE-FOLEY ANIMATION

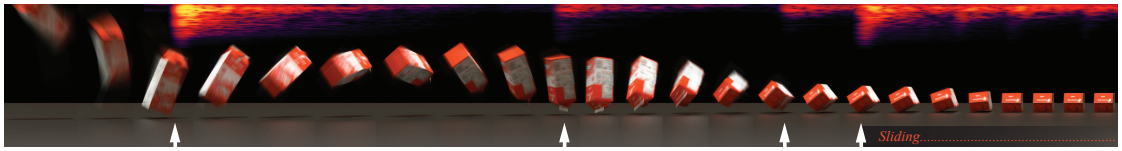


Figure 2.1: **Rigid-body motion from sound:** Given a recording of contact sounds produced by passive object dynamics, Inverse Foley Animation can estimate plausible rigid-body motions that have similar contact events occuring at similar times (Carton example).

In this chapter, we introduce *Inverse-Foley Animation*, a technique for optimizing rigid-body animations so that contact events are synchronized with input sound events. A precomputed database of randomly sampled rigid-body contact events is used to build a contact-event graph, which can be searched to determine a plausible sequence of contact events synchronized with the input sound's events. To more easily find motions with matching contact times, we allow transitions between simulated contact events using a motion blending formulation based on modified contact impulses. We fine tune synchronization by slightly retiming ballistic motions. Given a sound, our system can synthesize synchronized motions using graphs built with hundreds of thousands of precomputed motions, and millions of contact events. Our system is easy to use, and has been used to plan motions for hundreds of sounds, and dozens of rigid-body models.

2.1 Introduction

Synchresis: "... the spontaneous and irresistible mental fusion, completely free of any logic, that happens between a sound and a visual when these

occur at exactly the same time." [38]

If you hear contact sounds of an object bouncing around on the floor, you can probably create a plausible picture in your mind of what is happening. Unfortunately, computers and robots do not know how to hallucinate such motions just from the sound alone. In this work, we explore how contact sounds can be used to automatically synthesize plausible synchronized animations. Beyond pure intellectual curiosity, there are several reasons for doing so. Applications include low-cost sound-based motion capture, where plausible motion can be estimated or designed using sound alone. Sound can also help improve video-based estimation of rigid-body motion by providing contact event times, especially for blurry images of fast moving or small objects. In computer animation and games, synchronized contact sounds are commonly added "after the fact," by hand, or with event-triggered sound clips, or using digital sound synthesis. Unfortunately, manually adding sound to complex animated phenomena can degrade audiovisual synchronization, especially for rigid body motion, where mis-synchronized sound and impact events can be apparent. In contrast, digital sound synthesis can compute synchronized sound directly from the physics-based computer animation, but it can be difficult to achieve the realism of recorded sounds for many common objects. In this paper, we explore an alternate computational way to ensure audio-visual synchronization for physically based animation, while retaining the richness of pre-recorded and custom-designed sound effects.

Our technique, *Inverse-Foley Animation* (IFA), optimizes rigid-body animations to synchronize contact events with input sound events. We explore this sound-constrained motion design problem in the context of passive rigid-body

dynamics (see Figure 2.1).

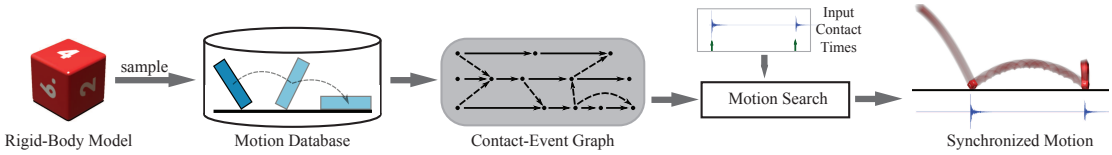


Figure 2.2: **Overview**

Given an input recording of contact sounds produced by a single object, a user indentifies contact event times when simulated rigid-body contact events likely occur. A virtual rigid-body dynamics model is constructed that approximates the scenario in which the sound was recorded. Since it is very difficult, if not impossible, to optimize a rigid-body simulation's parameters (initial conditions, restitution, friction, geometry, etc.) to produce contact events at the exact same times, we use the following data-driven approach. First, stochastic sampling is used to generate a rigid-body motion database, which we interpret as a library of rigid-body contact events (nodes) connected by ballistic rigid-body motions (edges). To help synthesize plausible contact event sequences that match the desired event times, we construct a contact-event graph, and introduce additional transition edges between contact-event nodes with similar attributes, e.g., similar orientation and velocity. By precomputing motion on a plane, we are able to globally adjust position, and planar orientation of contact events, using suitable transformations. We blend from one post-impact trajectory to another by optimizing a small impulse correction to match the subsequent contact state, e.g., orientation. A cost function is used to penalize implausible motion transitions, and to better synchronize with the sound. Modest motion retiming is used to obtain perfect synchronization with the input sound. Our contact-event graph can be searched for plausible motion paths, and supports constraints so the final contact event occurs at a contact-event node which is a terminal resting state. Additional contact constraints can be also be introduced, such as to make

an object land in a particular orientation, or to match contact locations observed in a video capture (see Figure 2.14). An overview of our approach is shown in Figure 2.2.

Using our approach we were able to generate plausible rigid-body animations with realistic synchronized sound. Our system has successfully synthesized motions for dozens of objects (see Figure 2.13) and hundreds of sounds, many of which would be hard to synthesize sounds for digitally, e.g., a scruffy bulb of garlic.

Our technique also provides a new way for animators to use sound to design physics-based animations. Unlike in space-time keyframing or other motion control techniques, our method only requires the user to specify time constraints on the simulation. We explored examples of physics-based motions designed using nonphysical input signals, such as timings from music scores.

2.2 Related Work

Event-based sound techniques [151] are widely used in computer graphics to synchronize sound effects, such as "clicks," with animated contact events, and pre-recorded sounds are routinely added by sound designers, or generated by foley artists, to enhance contact events. Unfortunately, complex sounds, especially involving multiple contact events of bouncing, sliding, rolling, chattering, etc., can be difficult to synchronize with pre-generated visual events. Alternately, recent advances in physics-based sound synthesis have enabled the generation of physics-based animations with various synchronized contact sounds [165, 114, 32, 171, 35]. Unfortunately, despite perfect synchronization,

generating realistic physics-based sound models can be hard for natural sounds, such as a crumpled paper ball, or a thin plastic shell, hitting a specific wooden surface. In contrast, we leverage natural recorded sounds, and explore ways to optimize audiovisual synchronization using motion control.

Inverse-Foley Animation can be viewed as a spacetime optimization problem [167, 40] involving rigid-body motion laws and frictional contact physics, but devoid of other spatial constraints. Unfortunately this motion estimation problem is extremely underconstrained and mathematically ill-posed, which complicates convergence for nonlinear optimization methods. Prior methods for motion design have leveraged the ability of humans to interactively navigate the space-time solution space of rigid-body contact events to obtain desired motions [122]. Unfortunately, they do not appear suitable for navigating time-only constraints efficiently, and while their derivative-based search methods could be used to optimize a sequence of contact times, they are inherently local methods whereas future contacts are easily created/destroyed when optimizing initial conditions and contact force perturbations. Motion sketching has been proposed to help animators estimate plausible motions they desire [123], which exploits the sketch to provide a good initial guess to help nonlinear optimization methods converge. In contrast, Inverse-Foley Animation is essentially a time-based sketch, which lacks spatial information to help nonlinear optimization.

Random sampling techniques have been used to explore the space of initial conditions and other simulation parameters, that could produce desired outcomes [154]. Barzel et al. [16] introduced the idea of plausibility for animations, arguing that there can be many acceptable simulations. Markov chain Monte Carlo (MCMC) has been used to sample animations satisfying specified constraints [36]. Similar sampling methods can be used to optimize contact-event

times, however downsides are that optimization times can be long, and that some methods (such as MCMC) require extensive parameter tuning. In addition, we found that forward sampling methods have a hard time hitting all of the contact event times, necessitating frequent restarts, and that it can be very hard to find plausible contact events that lead to terminal (zero energy) contact states at the desired times using forward search. Sampling reverse-time rigid-body motion might be better suited to this latter case, however the ill-posed nature of reverse-time motion poses its own challenges [161], and sampling multiple time-based constraints is still difficult. Many-Worlds Browsing [160] allows users to efficiently explore large numbers of sampled rigid-body simulations using a ranking interface, interactive browsing methods, and user-guided adaptive contact-force sampling, but does not provide any specific tools for global optimization of motion synchronization to find those needle-in-a-haystack simulations. In contrast, given an input sound, our contact-event graph can search a contact-event database to estimate synchronized motions. By using blending and time warping, our system can find plausible solutions even when the time constraints are hard (or infeasible) to satisfy.

Our use of contact-event graphs are closely related to "motion graph" techniques used to animate characters using motion-capture databases [86, 91, 7]. Prior work did not explore motion graphs for single rigid bodies since their motion is easy to compute, and other methods exist for rigid-body control. In contrast, we consider a graph of rigid-body contact events to optimize contact times efficiently. Like motion graphs, we also exploit the planar nature of motions in our database to freely orient and position contact events on the plane. There are several differences between contact-event graphs and traditional motion graphs. Motion graphs typically transition between each frame, whereas

contact-event graphs transition between contact events, with edges representing rigid-body free flight. Also, motion graph techniques often rely on ensuring the graph is connected, so arbitrarily long streams of motion can be synthesized. In passive rigid-body motion, energy is lost during each contact, so we may not expect to revisit states, and arbitrarily long streams of motion are not possible without accumulating large transition errors.

Synchronization has also been studied in work on aligning animation with sound and music. Cardle et al. [31] analyzed input MIDI scores, and transformed keyframe animation curves to improve synchronization. Kim et al. [81] extracted motion beats from input motion data, used these motion beat examples as nodes in a movement transition graph, and traversed the graph to synthesize motion that aligned to an input MIDI signal. Lee and Lee [90] used a "music graph" and retiming techniques to modify background music and animation curves simultaneously. In constrast, our contact-event graph method is heavily constrained by rigid-body contact physics. We also use time-warping techniques [29] but only to adjust simulated contact-event times to better match an input sound track.

In computer vision, the estimation of ballistic rigid-body motions can be challenging for fast motions and nonsmooth trajectories due to contact [168], and physics-based models can improve motion tracking [53]. For example, Bhat et al. [23] estimate ballistic rigid-body motion from video using an optimization method that exploits the smoothness of free-flight motion to track translational and rotational motion. In contrast, we use sound to constrain contact event times of nonsmooth rigid-body motions, possibly with position-level contact constraints from video, but with different yet plausible free-flight motions.

2.3 Input Contact-Event Specification

Given an input sound or other signal (e.g., MIDI events) for which we wish to find a synchronized animation, we first identify the timing and amplitudes of "contact events" in the audio stream. Unfortunately, identifying contact events robustly and automatically for arbitrary contact sounds, or other creative inputs, is a tricky problem, and we therefore rely on the user to provide the following information to the motion synthesis system.

Contact-time signature of sound: The user provides the following contact-event attributes:

- the target contact event times, $\bar{t}_1, \ldots, \bar{t}_n$, associated with the approximate beginning of each of n significant contact events. Without loss of generality we assume that $\bar{t}_1 = 0$.
- the rough duration of each contact event, $\Delta\bar{t}_k$. "Discrete" contact events, such as bouncing, which have no significant contact duration are given a zero duration. Longer and more complex "continuous" sequences of contact events, such as sliding or rapid sequences of micro-collisions at the end of motions, are given a nonzero duration (see Figure 2.3).

These times and durations indicate when contact activity should occur in the animation, and are determined by the user listening to the sound, and marking it in a sound visualization tool, such as Audacity. We refer to the list of significant contact event times and their durations as the *contact-time signature*, $\bar{\mathcal{T}} = \{(\bar{t}_k, \Delta\bar{t}_k)\}_{k=1..n}$. In practice, it takes less than a minute to identify contact events for a given input sound. Please see Figure 2.4 for examples of annotated contact sounds used in our system.

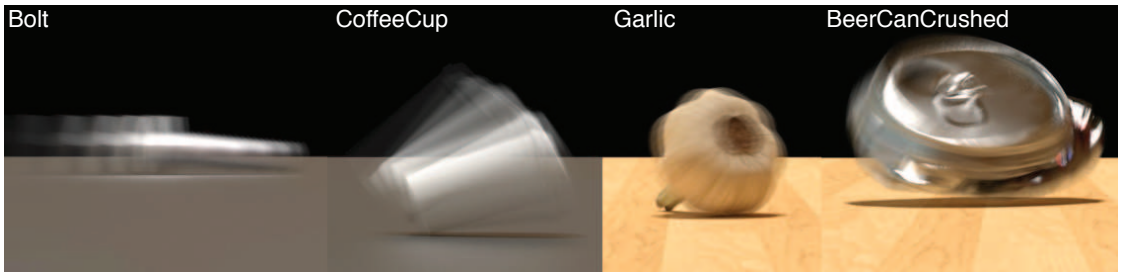


Figure 2.3: **Simulated continuous contact events synchronized to input sounds:** a sliding bolt, a chattering coffee cup, the scruffling of a rolling garlic bulb, and spolling of a crushed beer can.

The last contact event, n, is flagged as being a *terminal contact state* (i.e., a stable, zero-energy, rest state), or not. If not, then the synthesized motion can "bounce" out of the final contact event, instead of having to come to rest.

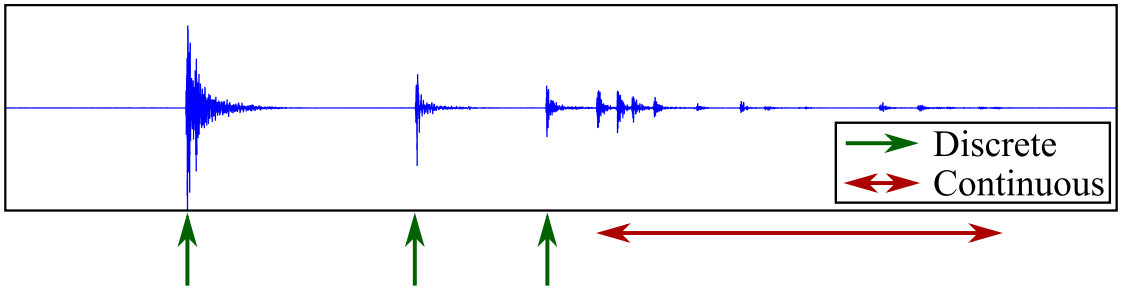


Figure 2.4: **Input sound** with user-annotated contact-event times.

Contact event amplitudes: Given the user-annotated sound signal, we can automatically estimate contact event amplitudes, $\bar{a}_1, \ldots, \bar{a}_n$, from the sound signal. For a discrete event at time $\bar{t}_k$, we set the amplitude $\bar{a}_k$ to be that of the nearest peak in a small time window—we use 20 ms in our setup. For continuous events, we use the amplitude values in the continuous region which the user selected. Assuming contact event k is s samples long, we refer to the amplitude in the j^{th} bin as $\bar{a}_k[j]$. To normalize the amplitudes, we divide through by $\bar{a}_1$, such that hereafter we assume that $\bar{a}_1 = 1$ (if the first event is continuous, we normalize by $\max \bar{a}_1[j], j = 1, \ldots, s$).

2.4 Rigid-Body Contact Problem

We seek to estimate motion for a single dynamic rigid body undergoing a sequence of contact events with a planar environment. In this section we define the rigid-body problem, notation, and its contact-time signature used to optimize synchronization.

Preliminaries: Rigid body dynamics is standard in computer graphics, and we refer the reader to an appropriate reference [19]. We will assume that the surface of the object, Γ, is approximated by a triangle mesh. The rigid-body motion is described by the following variables. The body has mass m, and body-space inertia tensor $\boldsymbol{I}$. Let the position of its center of mass be $\boldsymbol{x}$, its orientation given by the quaternion $\boldsymbol{q}$, the linear velocity by $\boldsymbol{v}$, and the angular velocity is denoted by $\boldsymbol{\omega}$. The rigid-body position is $\mathbf{p} = (\boldsymbol{x}, \boldsymbol{q})$, its velocity is $\mathbf{v} = (\boldsymbol{v}, \boldsymbol{\omega})$, and the total state is given by $\psi = (\mathbf{p}, \mathbf{v})$. Gravitational acceleration is $\boldsymbol{g}$. In our implementation, rigid-body motions were simulated using the Open Dynamics Engine (ODE), with contact modeled as a non-penetration constraint and solved as a velocity-level LCP. For planar contact, the convex-hull of each object model was used for rapid simulation.

Contact Events: For simplicity, assume that the environment is a single infinite plane, and that the rigid body undergoes passive motion under gravity that leads to sound-producing contact events with the plane (see Figure 2.5). We denote the times of these contact events by $t_1, t_2, \ldots, t_n$ for a sequence of n contact events, and again we are free to assume that $t_1 = 0$. These events may be either what we will call "discrete contact" events, such as bouncing impact, or longer "continuous contact" events, such as rolling, sliding, or rapid chattering. In either case, let t_k denote the starting time of the k^{th} event; denote the time immediately

before the contact event by t_{k-} and immediately after by t_{k+}. Denote the position at the start of the contact event by $\mathbf{p}_k$, the pre-impact velocity by $\mathbf{v}_{k-} = \mathbf{v}(t_{k-})$, and the post-impact velocity by $\mathbf{v}_{k+} = \mathbf{v}(t_{k+})$. For discrete contact events, such as bouncing, we can often assume without loss of generality that the event is instantaneous, i.e., $t_{k-} \approx t_{k+}$, however, in practice, the contact event may last one or more time steps depending on the type of integrator or contact resolution method used. In contrast, for continuous events, contact forces exist for a longer period of time, with $t_{k-} < t_{k+}$. We denote the rigid-body state immediately before impact by ψ_{k-} and the state immediately after impact by ψ_{k+}.

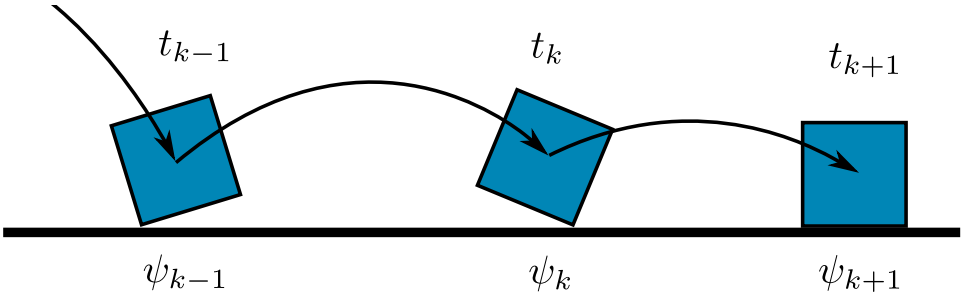


Figure 2.5: **Contact Sequence**

Contact-time signature of motion: In order to simplify comparison with other sounds and signals, we use a contact-time signature to summarize the rigid-body animation contact event sequence. Analogous to the contact-time signature for the input sound, $\bar{\mathcal{T}}$, we can define a contact signature for the simulation, $\mathcal{T}$, that consists of the contact start times, t_k, and the contact duration $\Delta t_k = t_{k+} - t_{k-}$. Ideally one could search the simulation space to find motions with matching contact-time signatures, $\mathcal{T} = \bar{\mathcal{T}}$. However, the raw simulation output is noisy, and requires some filtering before direct comparison to the simplified user-specified $\bar{\mathcal{T}}$. Without loss of generality, assume a fixed time step integrator, and consider the sequence of rigid-body contact impulses whose two-norm magnitude is given by $f_i \geq 0$ at time step $i = 1 \ldots N$. Based on this contact impulse data, we

can generate an initial noisy version of $\mathcal{T}$, which we then filter as follows (see Figure 2.6).

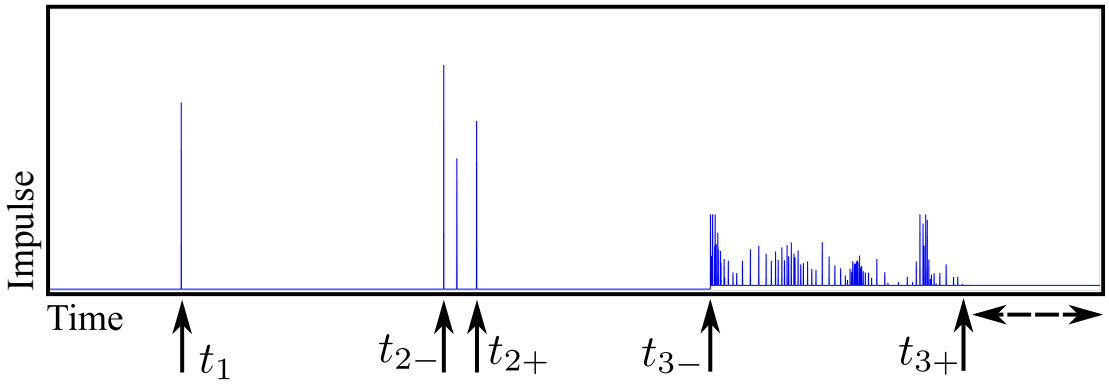


Figure 2.6: **Filtering the simulation's contact-time signature:** There is a discrete event at time t_1. Several close events are grouped into one event between times t_{2-} and t_{2+}. There is a continuous event between times t_{3-} and t_{3+}. The resting contact at the end, denoted by the dashed line, is clipped.

First, we filter $\mathcal{T}$ to remove small gaps between events by merging adjacent contact-event intervals that are separated by less than a small amount δt; in our implementation, we use $\delta t = 10$ ms. For example, if $[t_-, t_+]$ and $[t'_-, t'_+]$ are adjacent (with $t_* < t'_*$), we will merge them to form $[t_-, t'_+]$ iff $t'_- - t_+ < \delta t$. Contact events are renumbered accordingly after a merge. Note that this merging process creates longer "continuous" contact events that contain multiple micro-collision events.

Second, we filter $\mathcal{T}$ to detect discrete (short-lived) impact events. Contact events with duration below a specified duration δt are replaced by zero-duration discrete events: if $\Delta t_k < \delta t$ then we set $\Delta t_k = 0$. In our implementation, we used $\delta t = 10$ ms for most examples; however, we used $\delta t = 30$ ms for the BeerCan, Garlic, and Nut, to ensure the continuous sounds of those objects were treated as continuous events, and not broken up into many microevents.

Third, continuous contact events (with $\Delta t > 0$) can require additional filtering, since final resting states have nonzero contact impulses for which no contact

sound will be produced. We therefore detect resting contact states using a velocity criterion, and clip the final resting-contact time interval appropriately, i.e., $[t_{n-}, t_{n+}] \rightarrow [t_{n-}, t'_{n+}]$ for $t'_{n+} < t_{n+}$.

Finally, contact-impulse amplitudes a_k are accumulated along the way for contact events in $\mathcal{T}$. These are computed as the two-norm of all contact impulses f_i associated with each contact event. Given the importance of relative amplitude changes, we normalize all amplitude variations such that $a_1 = \bar{a}_1 = 1$. Note that simulated impulse amplitudes a_k and contact sound amplitudes $\bar{a}_k$ are very different quantities and can not be directly compared, e.g., to require "$a_k = \bar{a}_k$." Given the different nature of impulses and sound, these comparisons can only be used to very roughly specify when large or small impacts occur. Note that this normalization is always computed based on the first event in the motion path, which is not always the first event in the simulation (see § 2.5).

Synchronized motion problem: The central problem we solve is to find a plausible rigid-body motion which is synchronized with the input sound, in the sense that the sound's $\bar{\mathcal{T}}$ is equal (or close) to the simulation's $\mathcal{T}$, and that the relative amplitude variations are similar. Given that the first contact is always synchronized (since $\bar{t}_1 = t_1 = 0$ and $\bar{a}_1 = a_1 = 1$), we essentially have $n - 1$ remaining contact events to synchronize. The parameters to be optimized are the initial conditions of the initial impact ψ_{1-}, as well as small contact impulses that can perturb each contact event. While other simulation parameters are also uncertain, such as the rigid body's shape, mass, etc., as well as the contact friction and restitution properties, for simplicity we will assume that these are fixed and specified reasonably by the user.

2.5 Contact-Event Graph Approach

We use a data-driven approach to approximate solutions to the synchronized motion problem. Given the prevalence of flat surfaces in our environments, our method is optimized to find synchronized motions of a single rigid body in a planar environment.

2.5.1 Sampling Rigid-Body Motions

We construct a database of rigid-body contact events by simulating thousands of rigid-body contact sequences as follows. We randomly sample the pre-contact state of the object, $(\boldsymbol{x}_{1-}, \boldsymbol{q}_{1-}, \boldsymbol{v}_{1-}, \boldsymbol{\omega}_{1-})$, at the time of first contact. Given translation invariance of the planar environment, it is sufficient to select any center-of-mass position $\boldsymbol{x}_{1-}$ above the contact plane. We then sample a random quaternion orientation $\boldsymbol{q}_{1-}$ [87] (Figure 2.7a). The object is then pushed down into contact with the plane by finding the minimum-height vertex on the object (Figure 2.7b). The pre-contact velocities are sampled as follows (Figure 2.7c). The direction of the linear velocity $\boldsymbol{v}_{1-}$ is importance sampled from the lower hemisphere (so that the velocity is into the plane) with a cosine distribution about the normal used to emphasize more vertical directions[1]; the magnitude $\|\boldsymbol{v}_{1-}\|_2$ is uniformly sampled (we use $[0.5, 4]$ m/s). Angular velocities are nonuniformly sampled from within a 3-ball; we sample a random direction $\hat{\boldsymbol{\omega}_{1-}}$, then uniformly sample the magnitude (we use $[0, 20]$ rad/s) so as to bias the magnitude toward zero to avoid too many motions with rapidly spinning initial conditions. Using these initial pre-

[1]For most example objects, dropping was the natural motion, so we only sampled the lower hemisphere up to 30° from vertical. For the Nut and Bolt, where rolling motions were common, we sampled up to 80° from vertical.

contact conditions, the simulation is run forward until the object comes to rest (Figure 2.7d). To obtain pre-contact ballistic motion to animate motion for $t < t_1$, we integrate the initial state backwards in time until an earlier contact is found (Figure 2.7e); we simply forward integrate the rigid-body with negated-velocity initial conditions, $(\mathbf{p}_1, -\mathbf{v}_{1-})$. This process is repeated as many times as necessary to create a database of simulations. Note that given translational invariance, this motion sampling problem has 3 rotation and 6 velocity degrees of freedom; further exploiting planar rotational invariance yields an 8-dimensional sampling problem.

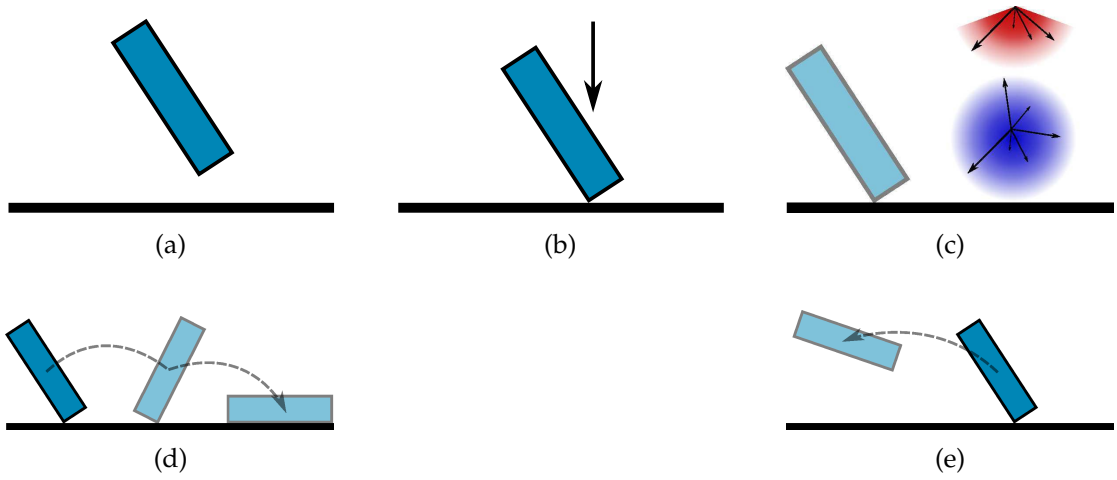


Figure 2.7: **Motion Sampling:** To sample initial simulation parameters, we (a) sample a random orientation, (b) push the object into contact, (c) sample linear (red) and angular (blue) velocities, (d) simulate forwards until the object comes to rest, and (e) simulate backwards to obtain pre-contact ballistic motion.

2.5.2 Contact-Event Graph Construction

The motion database is turned into a contact-event graph where each node represents a contact event, and edges represent inter-contact motions that transition between these contact states (see Figure 2.8). The weight on each edge represents

how expensive each transition is, which measures both rigid-body contact state errors, as well as synchronization errors when used for a specific contact-event time.

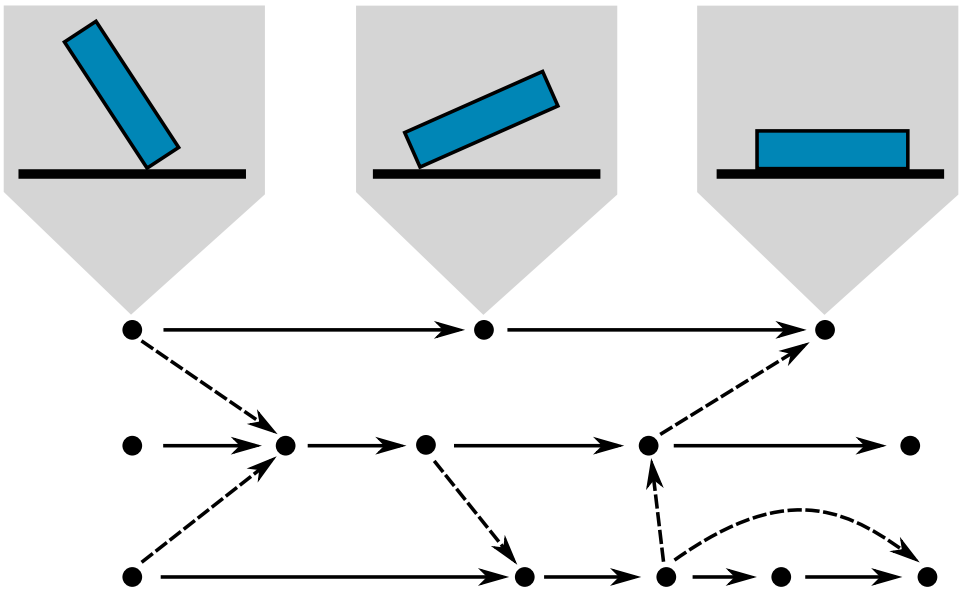


Figure 2.8: **Contact nodes and edges:** Nodes represent contact states, and edges represent transitions between these states. Solid arrows are physically simulated transitions, and dotted arrows are transitions we can make by computing a motion connection.

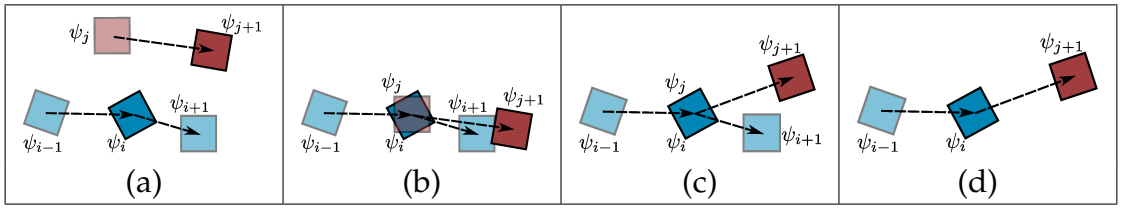


Figure 2.9: **Contact Registration (top down view):** (a) To transition from state ψ_i to state ψ_{j+1}, we register state ψ_j to state ψ_i. (b) First a planar translation is applied to align ψ_j with ψ_i. (c) Then a planar rotation is applied to minimize the orientation error between ψ_i and ψ_j. (d) Then we can evaluate the transition from ψ_i to ψ_{j+1}.

For each of the randomly simulated motions, we compute the filtered contact-time signature, $\mathcal{T}$, associated contact states (see §2.4), and contact amplitudes.

Each rigid-body simulation is abstracted as a sequence of contact-event nodes connected by edges which represent connecting trajectories. *Terminal nodes* represent contact states that bring the object to rest, and they are used exclusively to synchronize with terminal events in the input contact-time signature.

2.5.3 Motion Transitions

Given two motions which have similar contact events (similar post-contact velocity, orientation, etc.), we now describe how to smoothly transition from one motion to the other (see Figure 2.10).

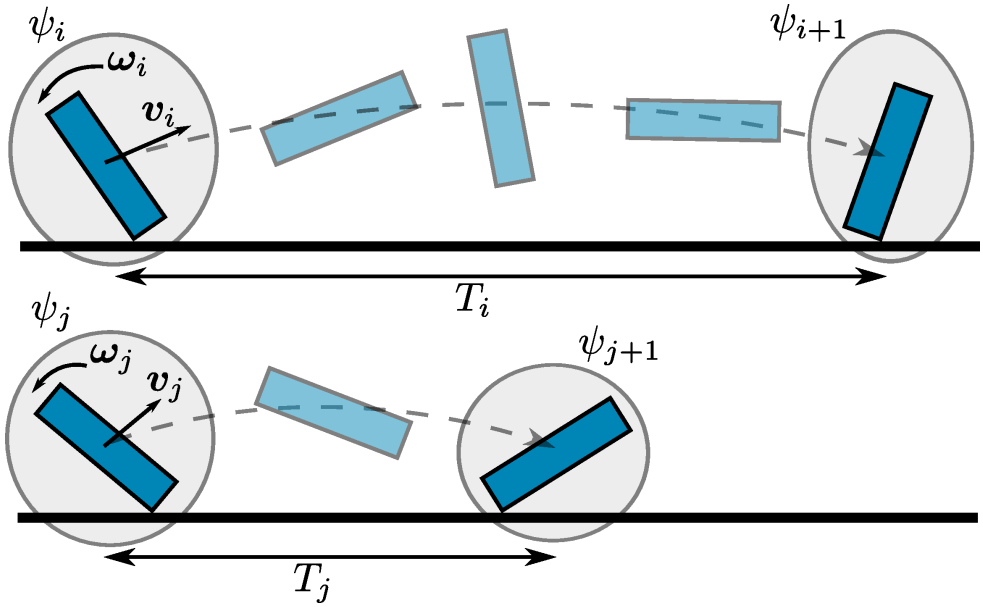


Figure 2.10: **Transitions:** Given two simulated contact sequences $\psi_i \rightarrow \psi_{i+1}$ and $\psi_j \rightarrow \psi_{j+1}$, we can transition from ψ_i to ψ_{j+1} by registering ψ_j to ψ_i and computing a motion connection.

Registering contact events

To evaluate contact state similarity (for edge weight determination) or to evaluate a motion transition, we exploit translational and rotational invariance on the plane to rigidly register contact events, thereby increasing the fit quality. To register two contact states i and j, we can transform state j to better match i by performing a planar registration of the two contact states at the post-contact times, t_{i+} and t_{j+} (see Figure 2.9). The rigid transformation involves (1) a 2D translation that best aligns the center of masses of the two bodies, and (2) a planar rotation that best aligns the orientations, $\boldsymbol{q}_{i+}$ and $\boldsymbol{q}_{j+}$. There is an analytical formula for this rotation [140].

Motion Connections

Given two similar contact states i and j, which we assume are registered, we can smoothly transition from state i to state $j+1$ by computing a modified rigid-body trajectory that connects the states. However, unless these two motions are *very* similar, directly interpolating the rigid-body trajectories through time, e.g., using rotational slerp or other methods for SE(3) interpolation [18, 68], can introduce nonphysical distortion for motions with very large and/or different angular velocities. Instead, we propose a different approach that involves computing a perturbation to the i^{th} post-contact momentum, so that the resulting motion matches the (registered) position and orientation at $j+1$. In our implementation, we compute momentum perturbations separately for the linear and angular components given the over-constrained nature of the boundary value problem.

Given two registered discrete-event contact states, i and j, we evaluate the

ballistic trajectory connecting i to $k = j+1$ by evaluating their linear and angular motions as follows. Without loss of generality, assume that contact events i and j both end at time $t = 0$, and impact k occurs at time T later. In our system we set $T = T_j$.

Linear Motion Connections: Linear blending is relatively simple, and is performed by computing the post-impact linear velocity required for the object's center of mass, starting at $\boldsymbol{x}_{i+}$, to end up at $\boldsymbol{x}_{k-}$ at time T later. It can be computed directly as $\boldsymbol{v}_{blend} = (\boldsymbol{x}_{k-} - \boldsymbol{x}_{i+})/T - T\boldsymbol{g}/2$, and is simply a modified v_{i+}.

Angular Motion Connections: The angular motion connection is slightly more involved. Denote the post-contact angular momentum (which is preserved over the ballistic trajectory) by $\boldsymbol{L}_{i+}$ and $\boldsymbol{L}_{j+}$, respectively. We seek to find an angular motion trajectory that has initial orientation $\boldsymbol{q}_{i+}$ at $t = 0$, and attains the final orientation $\boldsymbol{q}_{k-}$ at $t = T$. There are many possible solutions to this boundary value problem, and we use a forward shooting method that starts at $\boldsymbol{q}_{i+}$, then seeks to hit $\boldsymbol{q}_{k-}$ while keeping the initial angular momentum close to that of the original motions. Specifically, we seek a free-flight rigid-body angular motion that has an initial angular momentum near the average of the two momenta, $\boldsymbol{L} = (\boldsymbol{L}_{i+} + \boldsymbol{L}_{k-})/2 + \Delta\boldsymbol{L}$, where $\Delta\boldsymbol{L}$ is a correction, such that $\|\Delta\boldsymbol{L}\|^2_{I^{-1}} = \Delta\boldsymbol{L}^T\boldsymbol{I}^{-1}\Delta\boldsymbol{L} = \Delta\boldsymbol{\omega}^T\boldsymbol{I}\Delta\boldsymbol{\omega}$ is ideally small. We define an endpoint orientation error using the quaternion error function $f(\Delta\boldsymbol{L}) = |\log\left(q(T, \Delta\boldsymbol{L})q_1^{-1}\right)|^2$ where $q(T, \Delta\boldsymbol{L})$ is computed by integrating Euler's angular equations of motion for the rigid body. By linearizing $f(\Delta\boldsymbol{L}) \approx f(0) + \boldsymbol{J}\ \Delta\boldsymbol{L}$, where $\boldsymbol{J} = \frac{\partial f}{\partial \Delta\boldsymbol{L}} \in R^{1\times 3}$ is the Jacobian (which we compute using forward differences), we can write the Newton's method update (with weighting) as $\Delta\boldsymbol{L} = W\boldsymbol{J}^T\boldsymbol{\lambda}$, then determine $\boldsymbol{\lambda}$ so that $f = 0$, and thus obtain $\Delta\boldsymbol{L} = -W\boldsymbol{J}^T(\boldsymbol{J}W\boldsymbol{J}^T)^{-1}f(0)$. The weighting matrix W is set to the inertia matrix $\boldsymbol{I}$, thus giving extra momentum to heavy axes. When

doing the Newton step, we damp the updates to avoid large angular changes which could lead to instabilities. We use a simple model to reason about angular changes, $\Delta\theta \approx \|\Delta\boldsymbol{\omega}\|T = \|\boldsymbol{I}^{-1}\Delta\boldsymbol{L}\|T$; in our implementation, the $\Delta\boldsymbol{L}$ update is scaled so that $\Delta\theta$ is less than 10^o. In rare cases where Newton's method does not converge, we simply discard the edge.

Time Warping: Any transition edge can be slightly re-timed to match the target inter-contact time T using time warping [29], however in practice only very limited retiming is used since this can lead to significant distortions. In practice we are able to achieve target times by keeping retiming effects less than 30%. Note that any retiming is done *after* blending to avoid retiming the equations of motion.

2.5.4 Edge weights for inter-contact transitions

Transition errors can be divided into two types: (1) static errors for motion, which include velocity and orientation differences; and (2) dynamic errors for sound synchronization, which include retiming, amplitude errors, and whether the event is terminal or not. Static errors depend only on the contact events from the database, whereas dynamic errors are input-sound related and depend on the target contact signature being matched.

Transition quality is measured by a factored edge weight model:

$$w = f_v f_q f_t f_s f_E f_{end} \tag{2.1}$$

where the affinity factors f_* correspond to velocity match (f_v), quaternion/orientation match (f_q), time warping to ensure synchronization (f_t), contact sound/force similarity (f_s), energy limiter (f_E), and terminal events (f_{end}). Higher

edge weights correspond to better transitions. We now describe these factors, and the tuned parameters used in all of our examples.

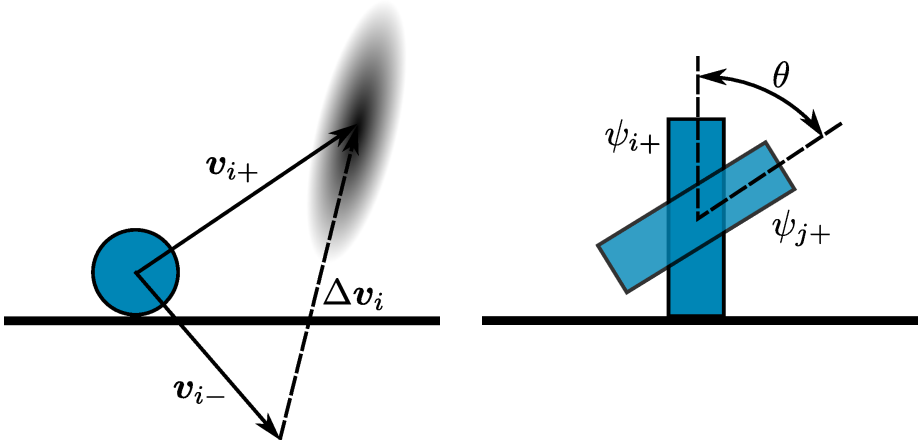


Figure 2.11: **Static state comparison:** (Left) The plausible region for velocities depends on the magnitude and direction of $\Delta\boldsymbol{v}_i$. (Right) Orientations are compared based on the smallest angle of rotation between the two states.

Post-contact velocity similarity is assessed based on the simulated velocity change $\Delta\boldsymbol{v}_i = \boldsymbol{v}_{i+} - \boldsymbol{v}_{i-}$ (see Figure 2.11), and $\Delta\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i+} - \boldsymbol{\omega}_{i-}$. We define plausibility parameters which represent the maximum plausible angle change for the linear (θ_L) and angular (θ_A) velocities, and the maximum plausible relative magnitude change for the linear (ϕ_L) and angular (ϕ_A) velocities. Inspired by O'Sullivan et al. [117], we use $\theta_L = 20^o, \phi_L = 0.4,$ and $\phi_A = 0.2$; for θ_A, we set $\theta_A = 10^o$. We evaluate the velocity similarity between two states using a weighted squared magnitude. We first compute rotations R_L and R_A, which rotate $\Delta\boldsymbol{v}_i$ and $\Delta\boldsymbol{\omega}_i$ to $(1, 0, 0)$, respectively, and compute

$$\mathbf{v}_{eval} = \begin{pmatrix} R_L(\boldsymbol{v}_{blend} - \boldsymbol{v}_{j+}) \\ R_A(\boldsymbol{\omega}_{i+} - \boldsymbol{\omega}_{j+}) \end{pmatrix}.$$

We evaluate the magnitude of $\mathbf{v}_{eval}$ using a scaling matrix which sets the magnitude to 1 at the edge of the aforementioned plausible region: defining the

diagonal matrix, $\boldsymbol{C}^{-1} = \text{diag}(a, b, b, c, d, d)$,

$$a = \phi_L \|\Delta \boldsymbol{v}_i\| \qquad b = \phi_L \|\Delta \boldsymbol{v}_i\| \tan(\theta_L)$$
$$c = \phi_A \|\Delta \boldsymbol{\omega}_i\| \qquad d = \phi_A \|\Delta \boldsymbol{\omega}_i\| \tan(\theta_A) \tag{2.2}$$

we set

$$f_v = e^{-\|\boldsymbol{C}\mathbf{v}_{eval}\|^2}. \tag{2.3}$$

The Gaussian halfwidth occurs at $\|\boldsymbol{C}\mathbf{v}_{eval}\|^2 \approx 0.69$, and so values outside our "plausibility" region ($\|\boldsymbol{C}\mathbf{v}_{eval}\|^2 > 1$) are unlikely.

Orientation error is important because it affects how much blending will be necessary to perform the transition. The exact plausibility parameter is difficult to quantify, due to the non-linearity of the equations of motion, as well as the fact that the amount of blending required also depends on the velocity similarity and length of the transition. The amount of acceptable orientation difference also depends on the length of the transition, since large differences for short transitions could require large connection impulses. We examine the similarity as a rate of angle difference per unit time. Letting $\theta = |\log(\boldsymbol{q}_i \boldsymbol{q}_j^{-1})|$, we set

$$f_q = e^{-c_q \max\left(\frac{\theta^2}{T^2}, \frac{\theta^2}{T_0^2}\right)} \tag{2.4}$$

where $c_q = 0.85$ and $T_0 = 0.2$ (this sets the Gaussian halfwidth to $\frac{\theta}{T} = 0.9$ when $T < T_0$, and limits the halfwidth to $\frac{\theta}{T_0} = 0.9$ when $T \geq T_0$, which is about 10^o).

Time warping improves synchronization, and its excessive use is penalized by the f_t factor. Given the goal duration of the inter-contact trajectory, T^*, and the blended trajectory time, T_b, we use

$$f_t = e^{-c_t T_{eval}^2} \quad \text{where} \quad T_{eval} = \left(\frac{\max(T^*, T_b)}{\min(T^*, T_b)} - 1\right), \tag{2.5}$$

with $c_t = 100$. This sets the Gaussian halfwidth to 7% retiming, and ensures that retiming errors beyond 15% are unlikely.

Target event sounds and contact forces are compared using 3 weights. Significant differences in relative amplitudes of the contact sound $\bar{a}$ and the simulated contact force a are penalized using the amplitude factor

$$f_a = e^{-c_a|\bar{a}-a|^2} \tag{2.6}$$

where $c_a = 2.78$, which sets the halfwidth to 0.5. The difference in continuous event lengths, Δt_1 and Δt_2, is measured using

$$f_l = e^{-c_\Delta|\Delta t_1 - \Delta t_2|^2}; \tag{2.7}$$

we use $c_\Delta = 7000$, setting the halfwidth to 0.01 seconds. To better synchronize amplitudes within continuous events, e.g., near the start or the end (see Figure 2.12), we match the centroid time of the sound event amplitude distribution, $a[k]$, (or contact force amplitude, $\bar{a}[k]$) as $\tau_a = (\sum_{k=1}^{n} a[k]k\Delta t)/(\sum_{j=1}^{n} a[j])$ (similarly, $\tau_{\bar{a}}$ for $\bar{a}[k]$). Given the event's sound and force centroid times, τ_a and $\tau_{\bar{a}}$, respectively, we compare them using the factor

$$f_c = e^{-c_\Delta|\tau_a - \tau_{\bar{a}}|^2}. \tag{2.8}$$

The sound comparison affinity factor is the product of the 3 weights,

$$f_s = f_a f_l f_c. \tag{2.9}$$

For discrete contact events (where $\Delta t \approx 0$), only f_a contributes.

Spurious energy gains are avoided by further restricting transitions. The energy of a state immediately before (-) or after (+) the i^{th} contact event is given by

$$E(\psi_i^{\pm}) = \frac{1}{2}\left(m\|\boldsymbol{v}_i^{\pm}\|^2 + (\boldsymbol{\omega}_i^{\pm})^T \boldsymbol{I}\boldsymbol{\omega}_i^{\pm}\right) + mgh_i^{\pm}$$

where $h_i^{\pm}$ is the height of the object's center of mass. To discourage transitions to

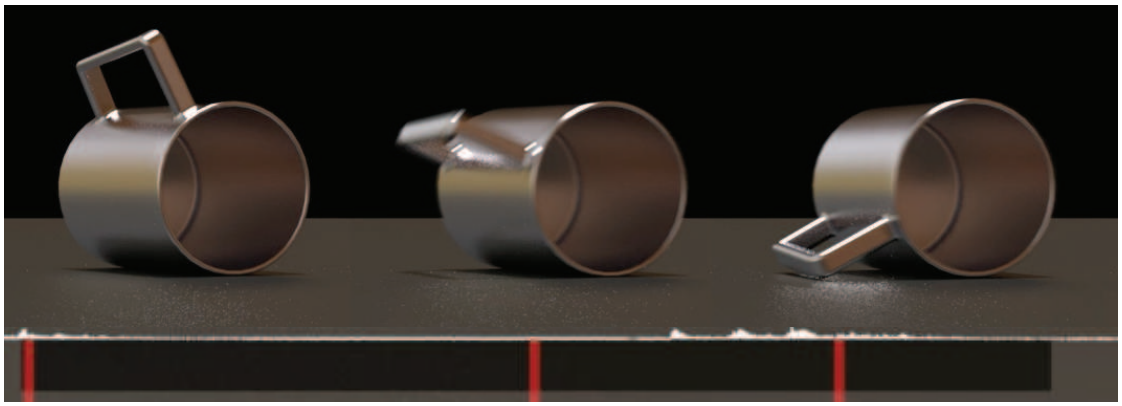


Figure 2.12: **A continuous contact event** (MetalCup desk 02): Using the sound amplitude (shown at bottom) and force amplitude centroids, we not only synchronize its initial impact (Left), but, as the mug rolls (Middle), also the later handle impact (Right).

higher energy states[2] we set

$$f_E = \begin{cases} 1, & E(\psi_i^-) > E(\psi_j^+), \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

Terminal nodes are used for the final contact event if the object is required to come to rest, and this is achieved by the f_{end} factor: if we are currently considering transitions from event m in the target sequence $\bar{\mathfrak{T}}$, then

$$f_{end} = \begin{cases} 1, & \text{if } (m \text{ not terminal}) \text{ XOR } (\text{state } j+1 \text{ is terminal}), \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

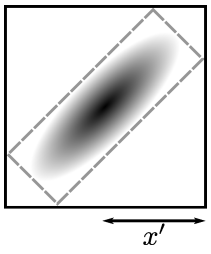
2.5.5 Searching for Synchronized Motions

Walks in the graph correspond to synthesized rigid-body contact sequences. Edge weights are defined such that larger edge weights correspond to good transitions. *We define the path quality (or score) as the product of the edge weights*, in-

[2]We found that requiring $E(\psi_i^+) > E(\psi_j^+)$ can be too restrictive, especially if the rigid-body model's restitution value is too low.

stead of the sum. Consequently a motion path can obtain a bad (near zero) score if there exists one rather implausible transition (with near zero edge weight). Given the large search space, we find walks in the graph using a branch and bound technique similar to [86], utilizing a k-d tree to quickly find the closest (best transition) events to the event we are currently at. If the input time signature has only one event, we can simply select a terminal event and its incoming motion. More generally, for $n > 1$ contact events, we find a feasible node synchronized with the second contact event at $\bar{t}_2$ by considering all events i, where T_{i-1} is close to $(\bar{t}_2 - \bar{t}_1)$. Subsequent contact events are found by considering subpaths of bounded search depth, e.g., the next 5 contact events, using a branch and bound technique. Edges are explored in order of weight, with the best ones being explored first. Given the best bounded-depth sub-path, we retain only the next two contact events, advance the contact-event search horizon, and repeat the process. As we approach the n^{th} terminal node, we restrict the search to include feasible subpaths, i.e., ones that have "terminal nodes" at the n^{th} contact event.

Finding nearest neighbors: We use two 12-dimensional k-d trees to quickly find potential transitions. One of the trees stores terminal events, the other stores non-terminal events. For an event j, the dimensions are $T_{j-1}, \Delta t_j, \mathbf{v}_{j-1+}$ (6 dimensions), the three angles $(\theta_1, \theta_2, \theta_3)$ that the rigid body's principal axes make with the contact normal, and the centroid time τ_j of the contact event. Denoting a point in the tree by $p = (p_1, \ldots, p_{12})$, we use a scaled L_2 distance when searching for nearby points: $dist(p, p') = \sum_{i=1}^{12} s_i^2 (p_i - p_i')^2$, with the scaling factors s_i set to $\frac{1}{halfwidth}$ of the corresponding edge weight affinity factor. When searching for neighbors of event j, the coordinate distance for the inter-



contact time (T) is normalized by T_{j-1}, and the principal axes distances are normalized by $\min(T_{j-1}, 0.2sec)$. The scaling of the velocities is slightly more involved because the plausible region is based on Δv_{j-1}, which is not always axis aligned. We first create a bounding box of the quadric described in (2.3). The box is centered at $\mathbf{0}$, and has edge half-lengths a, b, b (see (2.2)). The bounding box is rotated by R_L^{-1}, which aligns it with $\Delta \boldsymbol{v}_{j-1}$. The axis-aligned bounding box of the rotated original bounding box is computed, and has edge half-lengths x', y', z'. These half-lengths are then used to normalize the velocity distances. Angular velocities are treated similarly. Note that the k-d trees only need to be constructed once. The scaling in the distance function allows the distances to be adjusted depending on the query point.

Motion Score: The optimizer searches for paths that maximize the product of edge weights. To report a normalized motion score for humans (independent of the number of contacts), we define a motion's Score as the geometrically averaged affinity factor values over the n-contact motion sequence,

$$\text{Score} = \left(\prod_i f_v f_q f_t f_a f_l f_c \right)^{\frac{1}{6n}}. \tag{2.12}$$

To shed light on motion quality, we also report the Score independent of the sound amplitude factors,

$$\text{Score}_{\text{w/o sound}} = \left(\prod_i f_v f_q f_t \right)^{\frac{1}{3n}}. \tag{2.13}$$

Optional speedups: Since searching large contact-event graphs can be slow, we use several optional speedups. To avoid spending time exploring obviously poor transitions, we only explore edges where $\log(f_v) > -10$ and $\log(f_q) > -10$. Furthermore, we use the k-d tree to quickly find and search only the best 5 children/transitions of each node, thereby reducing the number of transitions

Figure 2.13: **Virtual models (left) and real-world objects (right)** used in over 434 IFA experiments.

that must be considered during the branch-and-bound search. For each recorded sound, we also use an "early exit" condition, that terminates the search (and returns the found motion) if the Score is sufficiently high; in our examples, we "early exit" if Score ≥ 0.3. We also enforced a maximum search time of 1 hour to time out on potentially infeasible problems.

Lazy evaluation of blends: We do not actually compute blends for edges during the graph search, to avoid the cost of computing blends for non-simulation edges in the graph, which is potentially very high, e.g., in graphs with hundreds

of thousands of edges can require many hours of Newton solves. In practice, we only require blends for edges used in the final animation. Some transitions can introduce ground interpenetration when blending sufficiently dissimilar motions. Since interpenetration can be perceptually bothersome it is not allowed. We initially assume that all edges are feasible, and then once we find an optimal sub-path we compute its blends, then if any edges are infeasible we discard them and recompute the optimal sub-path. In practice, blending costs are reduced enormously, and the search is still fast since very few edges are infeasible. We note that blends depend only on the motions, and not the input sound. While they could be precomputed, this would require much longer precomputation and more storage for the database.

2.6 Results

We recorded multiple sounds from 27 real-world rigid objects on multiple surfaces, for a total of 434 sound recordings. For each object, a virtual proxy model was created, with the geometry, mass properties, and contact restitution behavior set to produce a similar likeness. The objects and models are shown in Figure 2.13. Our system was able to synthesize plausible synchronized motions for all of the objects, and almost all of the sounds. Please see the accompanying video for all audiovisual results and Scores. Statistics are provided in Table 2.1; all timing statistics are run on a machine with four Intel Xeon 7560 processors at 2.27GHz. Note that *we use the same hand-tuned edge-weight parameters in all cases,* so no extra parameter tuning was required.

High-speed video comparisons: For some input contact sounds, we also

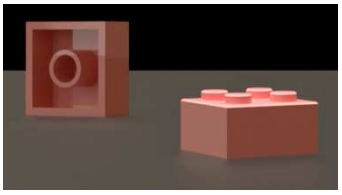
Rigid-body Model	nodes	n	success	search	Score
AdjustableWrench (desk)	2556981	5.5	13 / 13	0.9	0.43
BeerCanCrushed (wood)	1462089	2.9	15 / 15	0.6	0.35
Bolt (desk)	2604880	5.6	7 / 7	14.5	0.25
Carton (desk)	1282526	4.1	25 / 29	1.3	0.32
CoatHanger (desk)	4759056	6.4	18 / 19	9.3	0.46
CoffeeCup (desk)	5710615	7.1	15 / 16	38.2	0.24
CreditCard (desk)	1965526	4.4	22 / 23	6.8	0.44
d4 (desk)	3750746	15.7	6 / 10	62.7	0.17
d6 (desk)	6751058	20.4	13 / 14	12.1	0.41
d8 (desk)	4867818	22.0	6 / 12	53.9	0.27
d10 (desk)	7623610	25.7	11 / 14	18.6	0.38
d12 (desk)	9934621	26.0	4 / 7	24.2	0.39
d20 (desk)	9427739	26.8	9 / 10	37.7	0.31
Dustpan (wood)	4520601	6.7	6 / 6	21.9	0.37
Dustpan (carpet)	2629017	2.5	4 / 6	6.5	0.21
Eraser (desk)	2003606	4.6	47 / 48	2.1	0.50
FruitBasket (wood)	1697917	4.6	7 / 7	0.3	0.45
FruitBasket (carpet)	1705726	4.0	7 / 10	0.4	0.43
Garlic (wood)	876191	4.5	6 / 6	2.6	0.24
Keyboard (carpet)	2366495	5.9	8 / 9	11.4	0.30
Lego2x2 (desk)	4065133	9.3	9 / 9	16.3	0.36
Lego2x4 (desk)	3994256	8.0	23 / 23	8.8	0.41
MetalCup (desk)	4252157	12.8	6 / 6	22.1	0.32
MetalCup (carpet)	2378823	3.4	9 / 10	3.3	0.26
Nut (desk)	3790909	8.3	9 / 10	12.1	0.33
PaperBall (desk)	3160375	4.6	16 / 20	5.9	0.24
PlasticContainer (desk)	1797494	4.8	18 / 22	2.4	0.32
PlasticSpoon (desk)	1115891	4.6	10 / 15	2.6	0.38
TapeDispenser (wood)	2726872	6.4	10 / 10	1.8	0.41
TapeDispenser (desk)	3561017	7.9	17 / 17	10.8	0.36
Wrench (desk)	4862720	15.7	9 / 11	47.8	0.17

Table 2.1: **Statistics:** For each rigid-body model, a database of 400,000 rigid-body simulations were used to construct a graph with the indicated number of contact-event **nodes**. Multiple input sounds are used, with the average number of contact events (n). Motions were successfully estimated for most sounds, with the average search time (in *min*) for these successful cases reported (**search**) along with the average **Score**. Harder examples often having many contacts (such as the dice), longer search times, and a lower score. "Early exits" terminated searches that achieved **Score** > 0.3.

recorded high-speed video of the associated contact experiments (AdjustableWrench, CreditCard, Eraser, d6, Lego2x4, and PaperBall). Although the real and synthesized motions are different, these videos clearly demonstrate the level of synchronization attained. For some examples contact deformations are also visually apparent, e.g., the CreditCard.

Orientation constraints: Adding constraints to the search is helpful, especially for objects that have distinct sounds depending on the type of contact event. This is easily accomplished in our system by adding additional edge weights.

We demonstrate this by adding orientation constraints to the Lego, which has distinct sounds depending on whether its open side lands face down or not, and the Keyboard, which we always wanted to land with the keys facing up. Using the normal vector of a face, $\mathbf{n}$, and a goal direction, $\mathbf{d}$, we construct an affinity factor based on the angle between the two directions, $f_o = e^{-c_o\theta^2}$, where $\theta = \cos^{-1}\left(\frac{\mathbf{n}\cdot\mathbf{d}}{\|\mathbf{n}\|\|\mathbf{d}\|}\right)$, and $c_o = 140$ (which sets the halfwidth to $\theta = 0.0698$ rad $\approx 4^o$). For each contact node, if there is an orientation constraint for that node, we add the corresponding factor to the incoming edge weight. We matched all specified final orientations.



Position constraints and audio-visual capture: Another possible use case of IFA is approximate video-based motion capture. We captured several rigid-body motion sequences with a calibrated overhead camera, and labeled the approximate center of mass of the object at the start of each contact event, giving a desired position (in the plane) $\mathbf{b}_i$ at each contact event i. For simulated contact events, let $\boldsymbol{x}^p$ denote the position projected onto the plane. Motions with only one event could just be translated along the plane to match any position constraint. For

longer motions, we align $\boldsymbol{x}^p_{1\_}$ with $\mathbf{b}_1$, and rotate the motion to align the first edge ($\boldsymbol{x}^p_{2\_} - \boldsymbol{x}^p_{1\_}$) with ($\mathbf{b}_2 - \mathbf{b}_1$). For each subsequent edge, we add an affinity factor based on the position of each contact event, $f_p = e^{-c_p \|\boldsymbol{x}^p_{i-} - \mathbf{b}^p_i\|^2}$, with $c_p = 7000$ (which sets the halfwidth to 1cm). Audiovisual results are in the accompanying video and Figure 2.14, and error statistics are in Table 2.2.

Motion Example	**max (cm / relative)**	**average (cm / relative)**	L (cm)
Lego2x4 1	6.0 / .28	3.7 / .18	21.3
Lego2x4 2	7.4 / .18	4.1 / .1	41.1
Lego2x4 3	2.8 / .12	1.9 / .08	24.0
Garlic 1	8.3 / .09	4.4 / .05	96.8
Garlic 2	4.1 / .04	2.4 / .03	96.7
TapeDispenser 1	8.6 / .12	4.5 / .06	71.5
TapeDispenser 2	10 / .15	5.8 / .09	64.8
TapeDispenser 3	7.1 / .18	4.9 / .12	39.6

Table 2.2: **Position Constraint Statistics:** For each motion, we report the maximum and average position errors. Errors are given in cm, as well as units relative to the longest edge L of the motion's bounding box.

Music inputs: We also experimented with motion synthesis for highly nonphysical inputs taken from musical scores. Obtaining plausible synchronized motions can be challenging for such inputs due to infeasibility. For example, it is highly unlikely that a falling object would bounce indefinitely, or tap out a James Brown song. By relaxing plausibility and energy constraints, we were able to synchronize to music. We include two whimsical examples: (1) a 6-sided die tapping out the classic knocking rhythm "*Shave and a haircut. Two bits.*" (see video); and (2) we individually synthesized an ensemble of dice that tap to a popular drum track (see Figure 2.15). Terminal events were not imposed on any input contact-time signatures. To match longer input sequences, or ones with impossibly long pauses, we allow energy gains by modifing the energy growth factor f_E to allow a user-specified bounded energy gain.

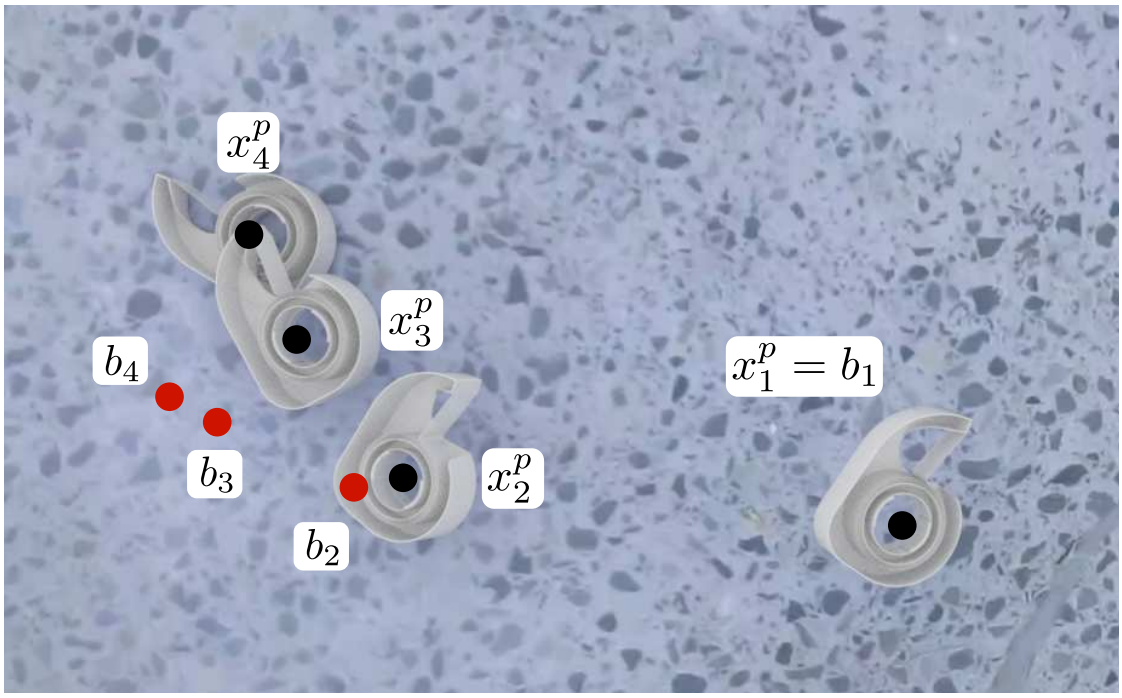


Figure 2.14: **Audio-visual motion estimation of rigid-body contact:** We estimate virtual rigid-body motions with planar object positions $\{\boldsymbol{x}_i^p\}$ (black dots) at contact events that approximate video-recorded object motions with indicated contact positions $\{\mathbf{b}_i\}$ (red dots).

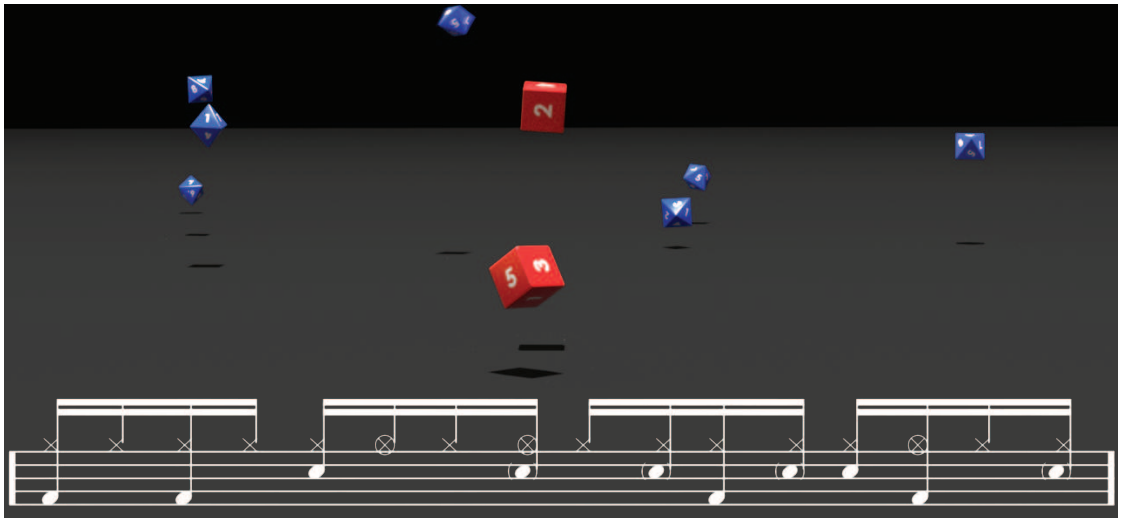


Figure 2.15: **Funky Dice:** Each die taps out drum tracks from the "Funky Drummer" rhythm; blue dice hit "high hat" sixteenth notes, and red dice hit snare and bass notes.

Database size dependence: We performed several experiments to evaluate the effect of the motion database size on the quality of the synthesized motion. For

four of our objects, the CoatHanger, d6, Eraser, and Lego2x2, we synchronized motions using different sized databases. For each database size, we ran the search for 1 hour, then returned the best motion. Results are shown in Figure 2.16. The motion quality tends to increase slowly with database size, although it is not monotonic due to the approximate nature of the search. Also note that for the d6, the search is unable to find a solution for the small database sizes.

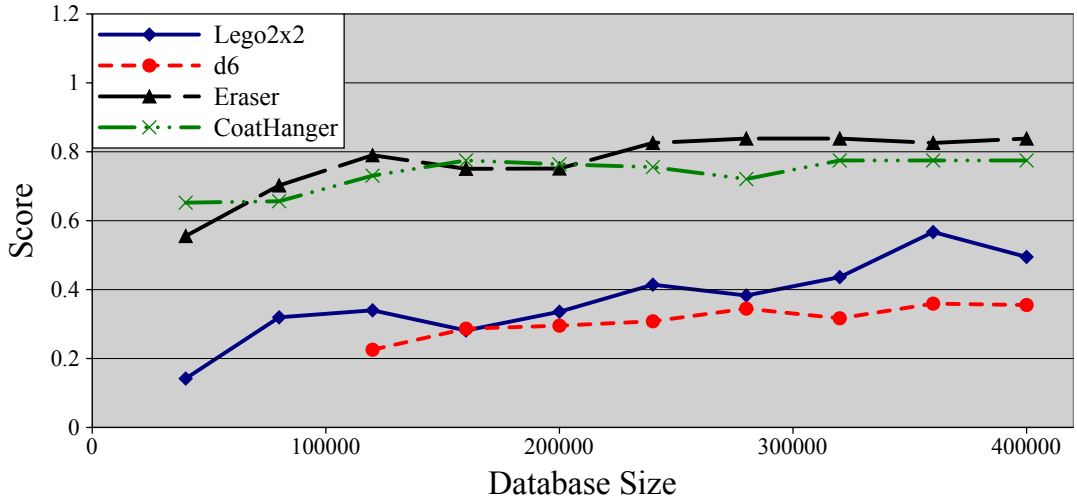


Figure 2.16: **Score vs Database Size:** We searched databases of varying sizes (# simulations) and compared the score (average factor weight) of the resulting motion (the same sound was used for all sizes). For each size, we ran the search for 1 hour (without any early stopping condition). For all four objects, the quality of the solution generally increases with database size. The d6 fails to find a solution when using small databases. The CoatHanger, Eraser, and Lego2x2 found a solution with all the sizes we tested.

2.7 Conclusion

We have introduced Inverse-Foley Animation, a new technique for synthesizing rigid-body motions that are synchronized with pre-recorded sounds, or other temporal input signals. By optimizing motion for synchronization, we

are able to capture the diversity and richness of real sounds, while avoiding the complexities and limitations of sound synthesis for such sounds. Inverse-Foley Animation has been successfully used to synthesize synchronized motions for dozens of objects, and hundreds of contact sound sequences. Furthermore, such digital techniques, if successful, have the potential to allow sound to be used earlier in the creative animation pipeline, potentially directing and improving the animated events, as opposed to having sound added "after the fact" in post production, and not achieving complete synchronization. For nonphysical input sounds, such as musical scores, IFA is a new tool for creative motion content generation.

Limitations and Future Work: Planning synchronized physics-based motions is particularly challenging. Perhaps the biggest limitation of our method is that, even for real-world contact sound inputs, it is not guaranteed to find a plausible solution. For a small fraction of the real-world contact-sound inputs, our system was not able to find a motion with a satifactory Score, which can be due to several factors: rigid-body dynamics modeling error (differing geometry, mass properties, friction, or (most commonly) contact restitution); violation of the rigid-body assumption, e.g., the CreditCard deforms noticeably on contact, can lead to different contact behavior; under-sampled motion databases can introduce infeasibility; even for large databases, the graph search method is not guaranteed to find solutions due to the incomplete nature of its branch-and-bound search—an issue for very long contact sequences, such as the dice (d4, d6, ..., d20). Other limitations arise from the fact that the input contact-time signature is a gross simplification of temporal contact dynamics, and hand labeling of contact times solely from sound can be ambiguous. For some of the sounds, the annotated contact signature may be sufficiently incorrect, and

impossible for the model to satisfy plausibly, even without rigid-body modeling limitations. Some contacts may be too quiet to hear, and fast contacts (e.g., when dice are chattering) may be too close to each other for humans to distinguish. Our system is not highly optimized for speed, and significant improvements could be made, e.g., to accelerate graph-based motion synthesis. Our approach has many motion design parameters, which we have hand tuned, and although we use the same parameters for our examples, better results may be obtained by optimizing them further. Finally, this work investigated the motion of a single rigid-body on a plane, and it is interesting to consider the more general problem of how to hallucinate animations that match sound. Future work should consider non-planar environments and multiple interacting objects, as well as other motion phenomena, such as characters, non-rigid bodies, and fluids, and other nonphysical temporal inputs. Finally, physics-based methods that use both audio and visual inputs streams are needed for tracking the motion of contacting rigid bodies.

CHAPTER 3

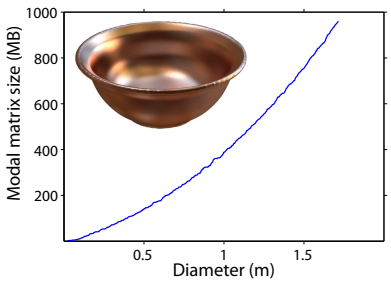
EIGENMODE COMPRESSION FOR MODAL SOUND MODELS

Figure 3.1: **Eigenmode Compression:** (Left) This complex Heptoroid model's displacement eigenmode matrix has 194 audible modes, 81884 vertices, and consumes 186 MB. By approximating each eigenmode with moving least squares (MLS), and nonlinearly optimizing the control points (shown in white), we compressed the entire model down to 3.1 MB—a 60:1 compression ratio—with *negligible* audible difference. (Middle) mode #17 (2.67 kHz, 276 MLS control points), (Right) mode #53 (5.21 kHz, 610 MLS control points).

In the previous chapter we demonstrated the power of simulation to generate a large amount of data. However, large amounts of data can be overwhelming. In this chapter, we propose and evaluate a method for significantly compressing modal sound models, thereby making them far more practical for audio-visual applications. The dense eigenmode matrix needed to compute the sound model's response to contact forces, can consume tens to thousands of megabytes depending on mesh resolution and mode count. Our eigenmode compression pipeline is based on nonlinear optimization of Moving Least Squares (MLS) approximations. Enhanced compression is achieved by exploiting symmetry both within and between eigenmodes, and by adaptively assigning per-mode error levels based on human perception of the far-field pressure amplitudes. Our method provides smooth eigenmode approximations, and efficient random access. We demonstrate that, in many cases, hundredfold compression ratios can be achieved without audible degradation of the rendered sound.

3.1 Introduction

3D modal sound models are one of the most effective and widely used techniques for rigid-body sound in animation and virtual environments. These methods benefit from the ability to precompute a solid object's eigenmode matrix and eigenfrequencies, to enable rapid integration of decoupled eigenmode vibrations, which are linearly composed for sound synthesis; since we do not hear object vibrations directly, acoustic transfer models can be precomputed to efficiently estimate sound pressure at the virtual listeners' positions. Previous decades have seen various advances to improve their speed, real-time performance, quality, measurement, contact dynamics, acoustic radiation modeling, all to generally make them faster and better.



Unfortunately, despite their success, high-quality 3D modal sound models still suffer from a serious, basic limitation: high memory requirements. Simply storing the precomputed displacement eigenmode matrix $\mathbf{U}$, a large dense N-by-M matrix for $N/3$ vertices and M modes, requires NM floats—tens to hundreds of megabytes even for small objects, dwarfing the size of other sound-model data, such as acoustic transfer. The problem is worse for larger and more complex objects, since they can have larger meshes (larger N) and more audible eigenmodes (larger M). For example, simply scaling up a large bronze bowl can quickly reach 1 GB of eigenmode storage (see inset). To make matters worse, future virtual worlds will involve not just one, but *many distinct* sound objects, each contributing to potentially massive memory overhead. Multi-gigabyte footprints are unacceptable and highly impractical for many applications. Unfortunately, without

knowing where the objects are hit, or how they are temporally driven, or where the listener will be, there is no principled way to *a priori* discard eigenmodes without potentially degrading the achievable sound quality.

To address this memory bottleneck without degradation to high-quality modal sound, we propose an eigenmode compression scheme that exploits the fact that (i) only sparse evaluation of eigenmodes is required, i.e., at contact locations, and (ii) larger relative errors are possible due to human hearing perception and varying radiation efficiency of eigenmodes. Given the eigenmode matrix, our automated compression pipeline builds an optimized Moving Least Squares (MLS) approximation for each eigenmode's smooth vertex displacement field (see Figure 3.1). Nonlinear optimization of MLS sample points and weights is used to minimize fitting error for a given number of sample points. To find the bounded-error approximation with smallest sample size, we also search over the number of MLS samples used.

We can achieve further compression by exploiting human perception of mode loudness. The main result we use is loudness dependence: humans' ability to distinguish amplitude differences decreases with quieter sounds [56], and perceived loudness is frequency dependent. By computing the expected loudness of each eigenmode (using an acoustic transfer model), we assign more error to quieter modes, thereby increasing compression without sacrificing quality. Importantly, assigning larger tolerable errors at very high frequencies (nearing the limits of human hearing) helps counteract the high geometric complexity of these rapidly oscillating modes.

To further increase compression, we exploit the fact that many man-made objects are symmetric, and their eigenmode matrix also possess symmetric struc-

ture. There are two types of symmetry we exploit (see Figure 3.2). First, in the case of intra-modal symmetry, we only need to store a small piece of a mode shape, and can use symmetry transforms to reconstruct the entire eigenmode from this piece. Second, some objects also exhibit inter-modal symmetries, where one mode is a rotation of another mode of the same eigenfrequency. In this case, we only need to store one mode of the pair.

Finally, we present results for numerous objects that demonstrate our method's ability to achieve high compression ratios without noticeable degradation (see Figure 3.3 and §3.7).

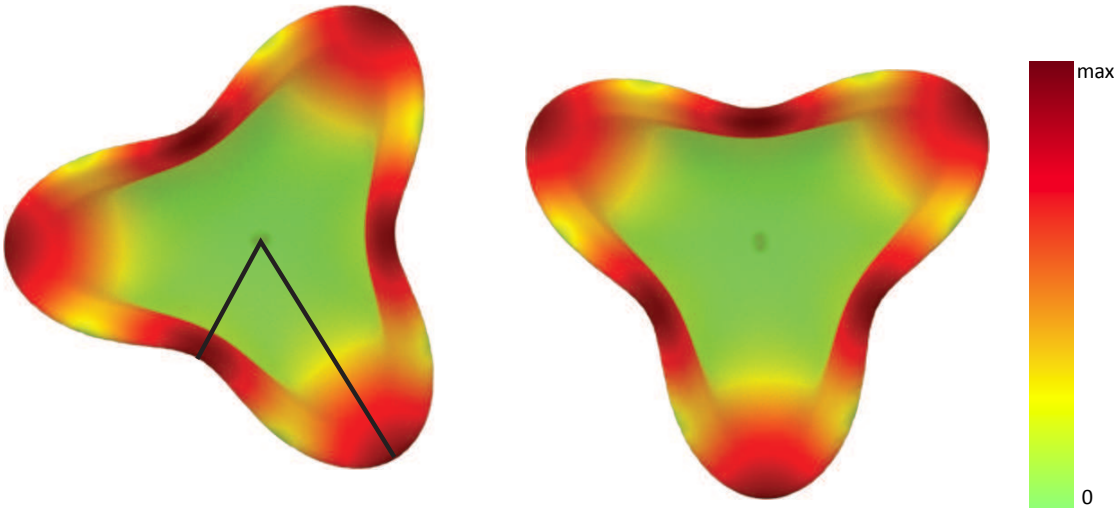
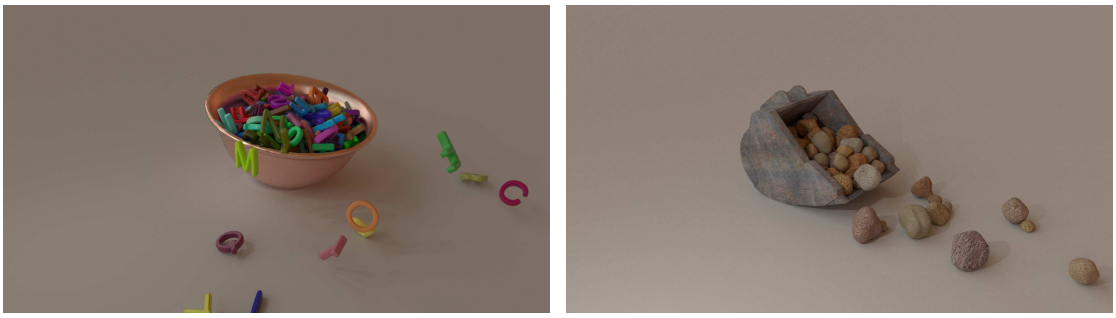


Figure 3.2: **Eigenmode Symmetries:** (Left) **Intra-mode or self symmetry** occurs when only a part of a mode (the indicated slice) must be represented, since the whole can be reproduced via symmetry transformations. (Right) **Inter-mode symmetry** occurs when a degenerate eigenmode is a transformed (here rotated) copy of another. Exploiting both symmetries here already gives 12× compression, and we further compress the slice. Figures are colored based on the magnitude of the eigenmodes, using the colormap shown. (Top down view of the Large Bowl model shown)

3.2 Related Work

Modal sound models are widely used in computer music and graphics, and modal vibration analyses are standard in engineering acoustics [138, 74]. Modal sound synthesis is a popular technique for producing plausible sounds of rigid objects [42, 162, 41], and a high-quality modal sound is a long-standing goal in computer sound [1]. Popularized for use in rigid-body computer animation over a decade ago [165, 115], many subsequent works make use of precomputed eigenmode matrices of 3D objects, e.g., using finite element analysis [115], for efficient runtime sound synthesis [75, 129, 25, 32, 171, 172, 35]. Unfortunately all of these approaches can suffer from large memory requirements unless small mesh resolutions, or limited accuracy sound approximations are used.

The high-cost of 3D physics-based modal sound models has been avoided in various ways in the past. Modal oscillator models can use simplified forcing models, without the need for spatial eigenmode data or finite element analyses [41]. Van den Doel and Pai [162] exploited modal "gain maps" based on analytical representations of eigenmodes, which avoid storage issues, but are restricted to special classes of resonators, e.g., planar membranes approximated by analytical solutions to the wave equation. Modal models and gain maps can also be estimated empirically from measured sound samples [165], and automated using robotic measurement [118] with adaptive sampling [132]. Arbitrarily high compression can be obtained by using highly simplified response models, e.g, in [165] the modal "gain map" for a wok is interpolated from just five measured points in [165]. In contrast, we attempt to enable high-resolution eigenmode representations for detailed sound models. Many techniques have been introduced to accelerate the modal sound synthesis by simplifying sound computations.



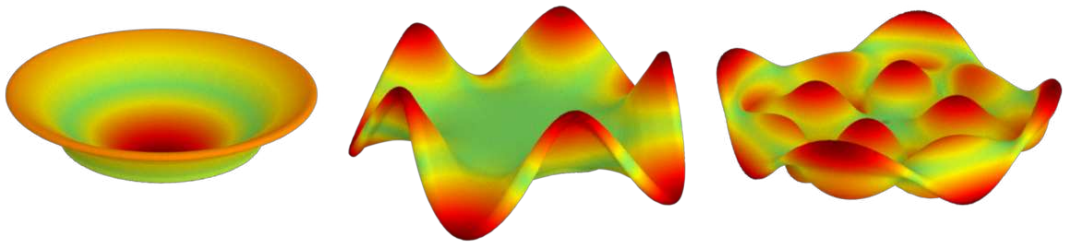
1.1 GB ⟶ 8.3 MB 676 MB ⟶ 13.7 MB

Figure 3.3: **Compression Benefits:** Eigenmode memory requirements for large simulations are drastically reduced without audible differences. (Left) 191 Letters are dropped into a large bowl. (Right) 125 Rocks fall out of a backhoe scoop.

Van den Doel et al. [164] use knowledge of human perception and auditory masking for runtime culling of modes which are either not sufficiently excited or audible. This accelerates runtime synthesis, but does not address the runtime memory costs of eigenmode matrices, since all modes must still be available in case they are not culled. Raghuvanshi and Lin [129] reduce eigenmode memory and synthesis costs by aggressively combining modes whose frequencies differ by less than the human frequency discrimination limit, achieving a space reduction factor of about 15. However, this approach can introduce large errors depending on where the object is hit, how it is driven, e.g., at a mode's resonance, or where the listener is positioned (when acoustic transfer is used). In contrast, we do not discard any modes (although runtime culling could still be used), and we achieve a much higher compression ratio, without introducing audible errors or subtle effects, e.g., like "beating" of nearby modes which can be perceptually important, such as for bells. Methods for auditory culling and spatial level-of-detail can reduce the number of sound sources at runtime [159], or aggressively cull modes [63], or delay or reduce mode simulation costs [25], but do not provide sound model compression.

Proxy sound models based on ellipsoidal soundbanks [171] can avoid the need to build and store object-specific modal models, however the soundbank can itself have an enormous memory footprint, which our method can compress dramatically. Instancing methods are often used to support sound synthesis with many objects, e.g., [25], simplying precomputation, but not avoiding the storage of at least one sound model.

High-quality sound pressure can require acoustic transfer functions, which add additional storage requirements of, e.g., multipole expansions [75, 171]. However, these costs are an order of magnitude less than for the eigenmode matrix in all of our experiments.



Mode 4, 2.6 kHz Mode 9, 4.9 kHz Mode 48, 16.6 kHz

Figure 3.4: **Linear Vibration Modes:** Three displacement eigenmodes $\mathbf{u}^j$ of the dinner plate model. We approximate the eigenmode values on the surface, which are needed to compute the system's response to external contact forces $\mathbf{f}$ via dot products, $(\mathbf{u}^j \bullet \mathbf{f})$.

We use MLS approximation, which has seen various applications in computer graphics for surface reconstruction [58, 144], defining implicit surfaces [3, 139], image deformation [135], and interpolating scalar fields [78]. We use MLS approximations and heavily optimize their control parameters, and exploit symmetries, to compactly represent M eigenmode vector fields per object.

Eigenmode compression is related to compressing collections of mesh displacement fields. Many mesh animation compression schemes use principal

components analysis (PCA) or other data reduction techniques to decorrelate the displacement fields [4], however these do not address how to compress the mode basis itself. Seo et al. [137] considered the problem of compressing blend shapes for animation, however their method exploits the sparsity of deformation fields which does not apply to global eigenmodes. Second-generation wavelets [136, 85] offer another approach to compressing surface displacement field operators, such as for "spiky" Green's function displacement fields to support fast wavelet summation [76], and have been used to compress parametrically coherent mesh animations [65]. However, they are less ideal for compressing pseudo-band-limited eigenmodes with sparse random access, especially when mode-specific mesh patches are needed to exploit symmetry.

There exists a large body of work in computer graphics on analyzing, detecting and processing symmetry [109]. One of the major applications of symmetry research is indeed compression [141], and many researchers have proposed different methods for analyzing and representing symmetric objects [96, 108]. Our approach builds upon the work of Martinet et al. [103], which we extend to detect geometric and modal symmetries simultaneously.

3.3 Background

Before explaining our compression approach, we briefly summarize the 3D modal sound model employed (which is identical to [171]), and discuss its memory requirements. We start with the equation of motion for a 3D linear elastodynamic model [17], $\mathbf{M}\,\ddot{\mathbf{u}}(t) + \mathbf{C}\,\dot{\mathbf{u}}(t) + \mathbf{K}\,\mathbf{u}(t) = \mathbf{f}(t)$ where $\mathbf{u} \in \mathbb{R}^N$ is the vector of $N/3$ nodal displacements, and $\mathbf{M}, \mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}, \mathbf{K} \in \mathbb{R}^{N\times N}$ are the mass, Rayleigh damping, and stiffness matrices, respectively; $\mathbf{f} \in \mathbb{R}^N$ represents any ex-

ternal nodal forces, such as contact forces, which act to excite vibrations. Modal analysis of linear finite element models is standard in graphics and engineering (see [138, 17]), and decouples the N equations of motion into independent simple harmonic oscillators. It amounts to first solving the generalized eigenvalue problems, $\mathbf{K}\,\mathbf{u}^j = \omega_j^2\,\mathbf{M}\,\mathbf{u}^j$, for the eigenvectors $\mathbf{u}^j$ and nonzero eigenvalues ω_j^2, with $j = 1 \ldots M$. Here $\mathbf{u}^j$ represents the j^{th} normal vibration mode's nodal displacement field, and ω_j is its natural frequency of vibration; M is the number of eigenmodes with frequencies within the audible frequency range of interest (20 Hz – 20 kHz). Some vibration modes are illustrated in Figure 3.4. The eigenmode matrix of interest is then

$$\mathbf{U} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^M \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{N \times M}$$

and can be normalized such that $\mathbf{U}^T\mathbf{M}\mathbf{U} = \mathbf{I}$. Letting $\mathbf{u}(t) = \mathbf{U}\,\mathbf{q}(t)$, where $\mathbf{q} \in \mathbb{R}^M$ are modal amplitudes, the equation of motion for the j^{th} mode becomes

$$\ddot{q}_j(t) + (\alpha + \beta\omega_j^2)\,\dot{q}_j(t) + \omega_j^2\,q_j(t) = \mathbf{u}^j \bullet \mathbf{f}(t) \tag{3.1}$$

and can be solved rapidly during sound synthesis [165]. Note that the eigenmodes $\mathbf{U}$ are needed at runtime to project the contact forces $\mathbf{f}$ into the modal subspace, i.e., via $\mathbf{U}^T\mathbf{f}$, and require NM floats of storage. For contact sound synthesis, one immediate source of compression is that only surface $\mathbf{U}$ values are needed, and so, for convenience (but a slight abuse of notation), we will hereafter assume that N refers to surface degrees of freedom.

The other significant storage requirement for high-quality modal sound is an acoustic transfer model to estimate sound pressure. Without loss of generality, the sound at position $\boldsymbol{x}$ and time t can be approximated as $\sum_{j=1}^{M} |p_j(\boldsymbol{x})|\ q_j(t)$,

where $p_j(\boldsymbol{x})$ is the acoustic pressure due to a unit-amplitude oscillation of mode j. Here it is approximated by the single-point multipole expansion

$$p_j(\boldsymbol{x}) = \sum_{\ell=0}^{\bar{n}_j} \sum_{m=-\ell}^{\ell} h_\ell^{(2)}(k_j r)\, Y_\ell^m(\theta, \phi)\, c_{\ell,m}^j, \tag{3.2}$$

where $k_j = \omega_j / c$ is the wavenumber, $h_\ell^{(2)}$ is a spherical hankel function of the second kind, and $c_{\ell,m}^j \in \mathbb{C}$ are the $(\bar{n}_j + 1)^2$ multipole expansion coefficients that must be precomputed and stored. For sufficient accuracy, the expansion order is set to $\bar{n} = \max(\frac{1}{4} k_j L, 4)$, where L is the diameter of the object [171, 97]. We note that, for all models we have analyzed, storage requirements for the uncompressed mesh-resolution-dependent eigenmode matrix, $\mathbf{U}$, are at least an order-of-magnitude larger than for the multipole coefficients, $\mathbf{c}$.

3.4 Eigenmode Approximation

We now describe how we fit each eigenmode $\mathbf{u}^j$ with an optimized Moving Least Squares (MLS) approximation, $\tilde{\mathbf{u}}^j$, such that its approximation error ε is below a mode-specific error bound, $\varepsilon(\tilde{\mathbf{u}}^j) \leq \varepsilon_{\text{goal}}^j$; we use a relative ℓ_2 error norm, $\varepsilon(\tilde{\mathbf{u}}) = \|\mathbf{u} - \tilde{\mathbf{u}}\|_2 / \|\mathbf{u}\|_2$. The MLS approximation has a number of desirable features for eigenmode approximation: it provides a smooth approximation suitable for approximating both low- and high-frequency eigenmodes of varying complexity; we can optimize the number of MLS samples and their parameters (positions and weights) to minimize fitting error while achieving high compression; it is a meshless approximation that accepts flexible input data, and simplifies implementation; it provides random-access eigenmode evaluation at O(1) cost at runtime; it can be adapted to exploit symmetry; and it efficiently interpolates samples across thin volumes. We note that this is an asymmetric compression

scheme: high compression is achieved using a computationally expensive MLS-fitting preprocess, but subsequent random-access decompression is fast. In later sections, we also describe how we gain further compression by: (1) when appropriate, we exploit intra and inter mode symmetry, and (2) we adjust each mode's error level $\varepsilon^j_{\text{goal}}$ based how well it radiates, as well as human frequency and amplitude perception.

3.4.1 Moving Least Squares

We seek to find a set of 3D control points $\mathbf{p} = (\mathbf{p}_i)_{i=1}^n$ with associated displacement values $\mathbf{w} = (\mathbf{w}_i)_{i=1}^n$ from which an MLS approximation can reconstruct the original mode accurately. Once we have them, we can evaluate a scalar component of the mode at $\boldsymbol{x}$, by first constructing an m-degree polynomial, $f(\boldsymbol{p} - \boldsymbol{x}) = \mathbf{c}^T \mathbf{b}(\boldsymbol{p} - \boldsymbol{x}) \in \Pi^3_m$, with $d = \frac{(3+m)!}{3!m!}$ coefficients $\mathbf{c} \in \mathbb{R}^d$, where $\mathbf{b}(\boldsymbol{p} - \boldsymbol{x}) \in \mathbb{R}^d$ is a vector of monomial basis functions. Given the control parameters $\mathbf{p}$ and $\mathbf{w}$, the coefficients $\mathbf{c}$ are computed by minimizing the MLS error,

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \sum_{i=1}^{n} \left[w_i - f(\mathbf{p}_i - \boldsymbol{x}) \right]^2 \theta(\mathbf{p}_i - \boldsymbol{x}). \tag{3.3}$$

Each xyz component of $\boldsymbol{f}$ is computed separately, by replacing w_i with $\mathbf{w}_{i,x}$, $\mathbf{w}_{i,y}$ or $\mathbf{w}_{i,z}$ in (3.3); we use a QR factorization to solve the least-squares problem, which involves solving with three (xyz) right-hand sides. Once $\boldsymbol{f}$ is fitted, the mode approximation at vertex $\boldsymbol{x}$ is simply $\boldsymbol{f}(\mathbf{0})$. The weighting function θ controls the influence of each control point; we use the adaptive $\theta(\boldsymbol{v}) = \exp(-\|\boldsymbol{v}\|^2/h^2)$ defined in [119], where $h = r/3$, with r the radius of the enclosing sphere of the k nearest neighbors of $\boldsymbol{x}$. This weight function allows the approximation to adapt to varying control point densities, and also improves performance since it is essentially zero for control points with $\|\mathbf{p}_j - \boldsymbol{x}\| > r$.

Storage and evaluation speed: Unless otherwise stated, we use cubic approximation, $m = 3$, and $k = 28$ nearest neighbors when choosing and optimizing control points. Storage of the MLS representation for an eigenmode with n_j control points requires $6n_j$ floats (3 for each p_i and w_i). The time required to approximate a mode value using the MLS representation is very fast, and still enables real time performance. In our implementation, evaluation takes about $40\mu s$, and involves finding the k nearest control points (using a kd-tree), and solving the 3 linear systems of size $k \times d$. Detailed timing statistics are given in §3.7.

Control point initialization using adaptive MLS: The key step in MLS compression is to choose where to place our n control points. We initialize their placement using the greedy selection method of [144]. Since we need at least d control points to ensure the linear system is not under-determined, we first randomly pick d mesh vertices as control points, setting their weights $\mathbf{w}$ to vertex displacements $\mathbf{u}$. We then iteratively improve the approximation by adding the vertex with the highest eigenmode error as the next control point, until the error reaches $\varepsilon_{\text{goal}}$ or the desired number of control points n is attained. This approach could quickly reach $\varepsilon_{\text{goal}}$ with a modest number of sample points ($\ll N$) for all examples we tested.

3.4.2 Control Point Optimization

By further optimizing the n control points and weights, we can significantly improve compression over adaptive MLS (see Figure 3.5). Since the MLS approximation, $\tilde{\mathbf{u}}$ is a function of the controls $\mathbf{p}, \mathbf{w} \in \mathbb{R}^{3n}$, we optimize their values

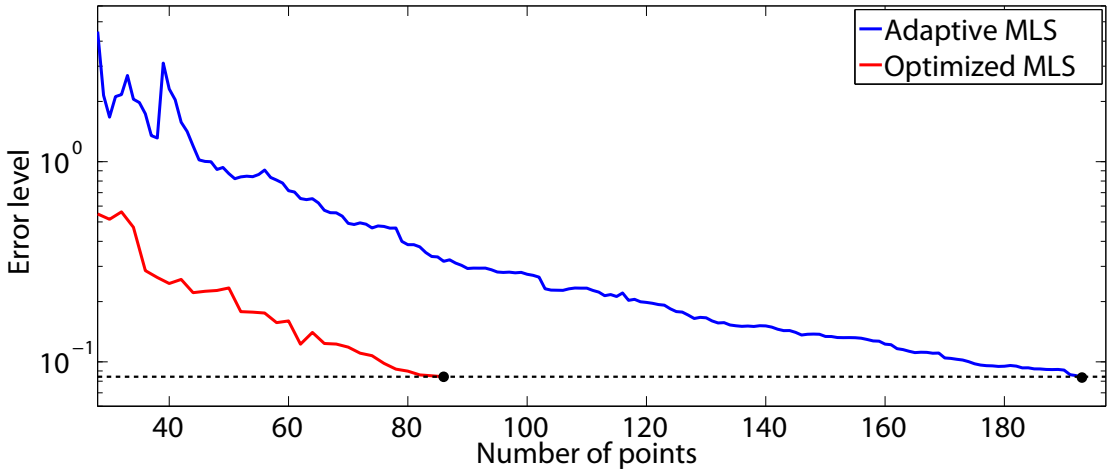


Figure 3.5: **MLS error convergence versus n:** Adaptive MLS provides fair compression at the target error, $\varepsilon_{\text{goal}} = 0.084$, but our optimized MLS fit requires even fewer control points (lower n).

using the nonlinear least-squares optimization of eigenmode error at vertices $\mathcal{V}$,

$$\min_{\mathbf{p},\mathbf{w}} \|\tilde{\mathbf{u}}(\mathbf{p},\mathbf{w}) - \mathbf{u}\|_2^2 = \min_{\mathbf{p},\mathbf{w}} \sum_{i\in\mathcal{V}} \|\tilde{\mathbf{u}}_i - \mathbf{u}_i\|_2^2. \tag{3.4}$$

We perform this nonlinear least-squares optimization using the Levenberg-Marquardt (LM) algorithm; we use the *Ceres Solver* implementation [2] which uses automatic differentiation to compute the Jacobian $\mathbf{J} = \nabla\tilde{\mathbf{u}}(\mathbf{p},\mathbf{w})$.

Multi-level optimization: Since the compressed models have very few control parameters, $6n$, the bottleneck of each LM iteration is not its linear solve, but rather the Jacobian computation which has cost dependent on the number of vertices, $|\mathcal{V}| \gg 6n$. To reduce this cost, we use a 3-stage multi-level approach: we first run LM to convergence using only $1/16$ of the vertices, $\mathcal{V}_{1/16}$, then improve this initial guess by running LM to convergence using $1/4$ of the vertices, $\mathcal{V}_{1/4}$, and finally we use all of the vertices, $\mathcal{V}$, as in (3.4). This dramatically reduces the number of expensive all-vertex Jacobian evaluations, since there are much fewer expensive LM iterations due to the improved starting guess. We use non-nested vertex sets, $\mathcal{V}_{1/16}$ and $\mathcal{V}_{1/4}$, where each is half randomly sampled, and half

importance sampled from $\|\mathbf{u}_i\|$ to avoid missing localized modes.

Minimizing n: Highest compression for mode j means finding the fewest control points, n_j, with bounded error, $\varepsilon(\tilde{\mathbf{u}}^j) \leq \varepsilon^j_{\text{goal}}$. Unfortunately the optimization process only adjusts the control points, $(\mathbf{p}, \mathbf{w})$, but cannot add or remove them. We use a simple binary search to determine the smallest n_j with $\varepsilon(\tilde{\mathbf{u}}^j) \leq \varepsilon^j_{\text{goal}}$, by repeatedly bisecting the minimum and maximum number of controls and optimizing them—the minimum n_j is initialized to $l_{\min} = d$, and the maximum $l_{\max}$ is taken as the number of greedily chosen adaptive MLS points. To save time on large examples, we "early exit" the expensive $\mathcal{V}$-level LM optimization when the error falls below $\varepsilon^j_{\text{goal}}$.

3.5 Exploiting Symmetry

Many objects, especially manufactured ones, are symmetric. If their modes also exhibit symmetries, we can increase the amount of compression by only storing parts of some modes, and using symmetry to reconstruct the rest. We first analyze the geometry to detect any cylindrical, n-way rotational, and mirror symmetries using the method of [103]. If approximate eigenmode symmetries exist, then we proceed to exploit them as follows.

3.5.1 Intra-mode symmetry

The first symmetry we exploit is intra-mode or self symmetry (see Figures 3.2 and 3.6). We slightly modify the geometric-symmetry method of [103] to detect object and eigenmode symmetries simultaneously. Instead of a purely geometric generalized moment function, we compute the *generalized geometry-eigenmode*

Mode 3, 1.9 kHz | Mode 15, 9.3 kHz | Mode 33, 16 kHz

Figure 3.6: **Intra-mode symmetry examples:** (Left) mirror symmetry; (Middle) 4-way rotational symmetry, plus several mirror symmetries; (Right) cylindrical symmetry.

moment function of order $2p$,

$$\mathcal{M}^{2p}(\hat{\boldsymbol{v}}) = \int_{\mathbf{s}\in S} \|\mathbf{s} \times \hat{\boldsymbol{v}}\|^{2p} \, \|\mathbf{u}(\mathbf{s})\|^{2p} \, \mathrm{d}\mathbf{s}, \tag{3.5}$$

where $\mathbf{s}$ is a vector from the surface's center of mass to a point on the surface, S. It follows that their real-valued spherical harmonic representation is given by

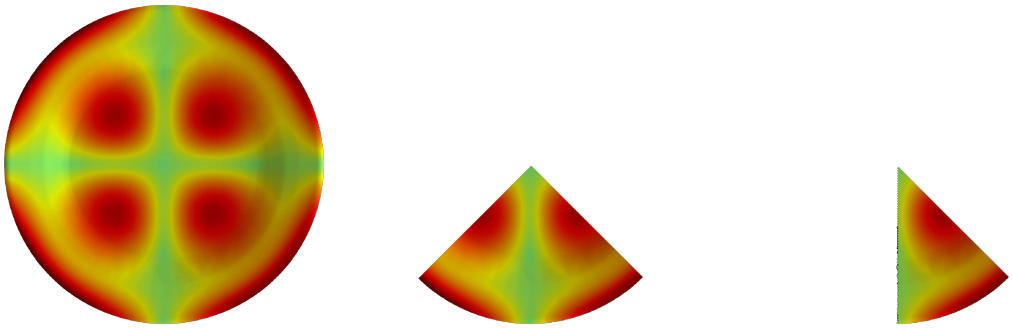
$$\mathcal{M}^{2p}(\hat{\boldsymbol{v}}) = \sum_{l=0}^{p} \sum_{m=-2l}^{2l} C_{2l,m}^{2p} \, Y_{2l}^{m}(\hat{\boldsymbol{v}}), \tag{3.6}$$

$$C_{2l,m}^{2p} = S_{p}^{l} \int_{\mathbf{s}\in S} \|\mathbf{s}\|^{2p} \, \|\mathbf{u}(\mathbf{s})\|^{2p} \, D_{2l}^{0,m}(R_{\mathbf{s}}) \, \mathrm{d}\mathbf{s}, \tag{3.7}$$

where formulae for S_{p}^{l} and $D_{2l}^{0,m}(R_{\mathbf{s}})$ are given in [103]. By searching among roots of $\nabla\mathcal{M}^{2p}(\hat{\boldsymbol{v}}) = 0$, we find candidate symmetry axes, and then classify the symmetries of the eigenmode magnitudes as either cylindrical, n-way rotation, or mirror symmetries as described in [103]. In our implementation, we use order $2p = 8$ moment functions.

Patch Extraction: After detecting symmetries using our generalized moment

function, we find a patch of the object to approximate with MLS for storage. First, the symmetries are sorted in order of their expected compression contribution: cylindrical symmetries, n-way rotational symmetries, and finally mirror symmetries. Then the symmetries are checked in order. If a symmetry works for the current set of vertices, only a patch of the vertices are MLS-compressed and stored, and the process continues with this patch. For cylindrical symmetries, we store a 5^o slice of the object's surface. For mirror symmetries we store one side of the mirror. For n-way rotational symmetries, we only need to store a $\frac{360^o}{n}$ slice. However, often there are orthogonal mirror symmetries to the n-way rotational symmetries which can provide additional savings. Therefore, when choosing which slice of an n-way symmetry to store, we search the remaining symmetries for any orthogonal mirrors, and position our slice to take advantage of one of them, if found (see Figure 3.7).



(a) (b) (c)

Figure 3.7: **Intra-mode symmetry example:** Starting with a full mode (a), we detect symmetries and only save a small patch of the mode. In (b), a 4-way rotational symmetry is used, and in (c), a mirror symmetry is exploited.

Displacement transformation: After finding a direction-invariant symmetry, $\|\mathbf{u}(\boldsymbol{x})\| \approx \|\mathbf{u}(\boldsymbol{R}\,\boldsymbol{x})\|$ for some $\boldsymbol{R}$ symmetry transformation, it still remains to determine any orthogonal transformation needed to match the eigenmode patch's vector displacements. We use a least-squares solve on vertex data to estimate

any (orthogonal) displacement transformation, $\boldsymbol{T}$ (with $\|\boldsymbol{T}\|_2 \approx 1$) such that $\mathbf{u}(\boldsymbol{x}) = \boldsymbol{T}\,\mathbf{u}(\boldsymbol{R}\,\boldsymbol{x})$.

Symmetry tolerances: Given the approximate nature of discrete eigenmode symmetry (due to meshing, MLS interpolation, numerical eigenanalysis, etc.) we use a tolerance when confirming eigenmode symmetry; in our results, we use 0.02.

3.5.2 Inter-mode symmetry

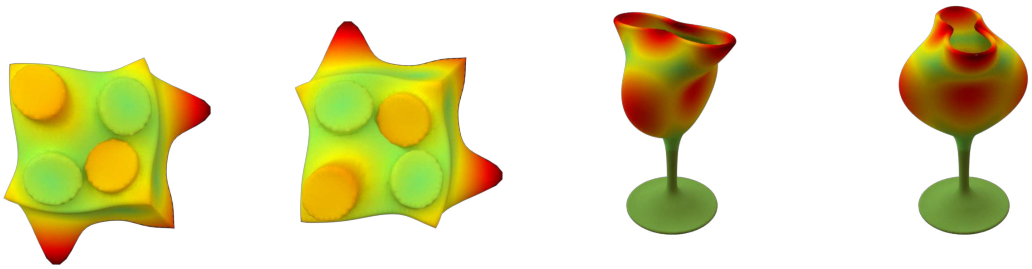
Beyond symmetry within a single mode, an interesting characteristic of cylindrically and n-way rotationally symmetric objects is that they can have degenerate eigenmodes, i.e., modes with near-equal eigen-frequencies, which form rotationally congruent pairs (see Figures 3.2 and 3.8). If we can detect a congruent pair (j, j'), we only need to store one of them along with the relative rotation which maps one to the other. We detect these pairs by summarizing the angular structure of the modes in a low-dimensional Fourier basis to find a candidate rotation, and then perform a rigorous verification of the candidate. Furthermore, we observe that congruent pairs are usually close to each other in frequency, so instead of doing this for all pairs (j, j'), we only do it for pairs such that $j' = j + 1$ (assuming modes are numbered in order of increasing frequency).

For a given pair of modes (j, j'), we first focus on the problem of finding a best rotation angle $\phi_{j,j'}$ about a known symmetry axis. For mode j, we compute Fourier-like moments,

$$a_m^j = \int_S \|\mathbf{u}^j(\boldsymbol{x})\|\, e^{im\phi(\boldsymbol{x})}\, dS_{\boldsymbol{x}}, \quad m = -\bar{m} \ldots \bar{m}$$

that describe the mode's amplitude variation about the rotation axis,

$$A^j(\phi) = \sum_{m=-\bar{m}}^{\bar{m}} a_m^j\, e^{-im\phi}.$$



Mode 10
7 kHz

Mode 11
7 kHz

Mode 19
11.78 kHz

Mode 20
11.78 kHz

Figure 3.8: **Inter-mode symmetry:** Pairs of rotationally congruent eigenmodes (shown here for Lego and Wine Glass models) just need to store one of the modes and a relative rotation.

In our implementation we use $\bar{m} = 30$. Much like the generalized moment function $\mathcal{M}$, this representation allows us to efficiently search for candidate rotation angles that map $\mathbf{u}^j$ to $\mathbf{u}^{j'}$. For two congruent modes (j, j') that have $A^j(\phi) = A^{j'}(\phi + \phi_{j,j'})$, it follows that $a_m^{j'} = a_m^j e^{im\phi_{a,b}}$. Therefore, we can estimate the rotation angle by considering the near-zero minima of

$$E(\phi) = \sum_{m=-\bar{m}}^{\bar{m}} \left| a_m^j - a_m^{j'} e^{-im\phi} \right|^2 .$$

Given the global minima of $E(\phi)$, we verify that the modes are congruent by evaluating the relative global error,

$$\int_S \left\| \mathbf{u}^j(\boldsymbol{x}) - \boldsymbol{R}_{\phi_{j,j'}}^T \mathbf{u}^{j'}(\boldsymbol{R}_{\phi_{j,j'}} \boldsymbol{x}) \right\| dS_{\boldsymbol{x}} \Big/ \int_S \left\| \mathbf{u}^j(\boldsymbol{x}) \right\| dS_{\boldsymbol{x}}, \tag{3.8}$$

where the look-up at the rotated position, $\mathbf{u}^{j'}(\boldsymbol{R}_{\phi_{j,j'}} \boldsymbol{x})$, is approximated using MLS on the mode's vertex data (using $m = 0$, $k = 3$). In this way, we consider all geometric cylindrical and n-way rotation symmetry axes, estimate the best rotation angle for each, and take the symmetry with the lowest feasible global error.

Symmetric Acoustic Transfer: One additional benefit of detecting congruent eigenmodes, is that their acoustic transfer functions are also congruent fields. Therefore it is only necessary to compute acoustic transfer multipole coefficients

for one of the modes, and the other one is obtained by rotating the multipole expansion coefficients using standard spherical harmonic techniques.

3.5.3 Runtime Evaluation

Given an external force $\mathbf{f}_i \in \mathbb{R}^3$ acting on node i at vertex position $\boldsymbol{x}$, we evaluate each eigenmode's value $\tilde{\mathbf{u}}^j(\boldsymbol{x})$, and evaluate the dot product $\tilde{\mathbf{u}}^j(\boldsymbol{x}) \bullet \mathbf{f}_i$ required to force each mode j. To evaluate a $\tilde{\mathbf{u}}(\boldsymbol{x})$ value (dropping the j superscript) for a mode with symmetry, we transform $\boldsymbol{x}$ to its symmetric image location, $\boldsymbol{x}'$, perform the MLS lookup $\tilde{\mathbf{u}}'(\boldsymbol{x}')$, then apply $\boldsymbol{T}$ to get the final $\tilde{\mathbf{u}}$.

Mode Caching: Simulations will often apply contact forces to the same vertex repeatedly, e.g., during sliding or resting contact. Consequently repeated reconstruction of identical mode values can ensue, which is wasteful. We therefore allow a small cache of recently reconstructed vertex mode values, and reuse them if they are repeatedly queried. For simulations with persistent and/or resting contacts we often observe a cache hit rate > 99%, whereas simulations dominated by rolling have very poor cache hit rates.

3.6 Perceptually Based Error Allocation

Choosing good MLS fitting error tolerances, $\varepsilon^j_{\text{goal}}$, are critical, as they control the balance between compression and quality. While one could simply set all $\varepsilon^j_{\text{goal}}$ to a conservatively small value which guarantees no audible difference will be heard (e.g., 2.3% would ensure an imperceptible 0.2 dB difference), much higher compression can be obtained (especially at high frequencies) without sacrificing quality by setting per-mode errors based on human perception of each mode's

sound. We incorporate three factors: (1) vastly different eigenmode loudness due to radiation efficiency, (2) frequency-dependent hearing sensitivity, and (3) a perceptual model of just noticeable differences (JND) in sound amplitudes.

We reason about the relative loudness of mode j by considering its far-field radiated power for a random excitation; we estimate a relative average pressure as proportional to $\|\mathbf{u}^j\| \, \|\mathbf{c}^j\|_F / (\omega^j)^2$ (see Appendix A.1 for a derivation). Since the average far-field pressure is only proportional to this value, we shift the dB pressure values by p_{dB}^{shift} so that the maximum modal pressure occurs at 60dB. After calculating the average pressure for a mode, we weight it with the ISO-226 equal-loudness curve (see Figure 3.9), which normalizes sound pressure levels at different frequencies to more accurately represent human hearing. Since this ISO standard is only defined up to 12.5 kHz, we extended it using data from [8]: we fit a cubic spline to interpolate the ISO data, and added three additional (frequency, SPL) points: (16 kHz, 41.8 dB), (18 kHz, 64 dB), (20 kHz, 89.9 dB).

The final representative modal pressure amplitude combining these factors is

$$p_{dB}^j = 20 \log \left(\frac{\|\mathbf{u}^j\| \, \|\mathbf{c}^j\|_F}{(\omega^j)^2} \right) - \text{ISO-226}(\omega^j) + p_{dB}^{shift}. \tag{3.9}$$

We then use this weighted average pressure to set a compression error level so that the rendered differences are expected to be near the JND. Studies in psychoacoustics have determined that humans' ability to distinguish amplitude differences in sounds varies with the sound's loudness, with larger differences tolerated at lower amplitudes [56] (see Figure 3.10). Given an acceptable pressure amplitude difference Δp (in dB) for a mode, the corresponding error is $\varepsilon_{\text{goal}} = 10^{\Delta p/20} - 1$. Based on JND amplitude differences Δp for a single tone at different amplitudes p_{dB} ([56], p.180), we model the acceptable error level using

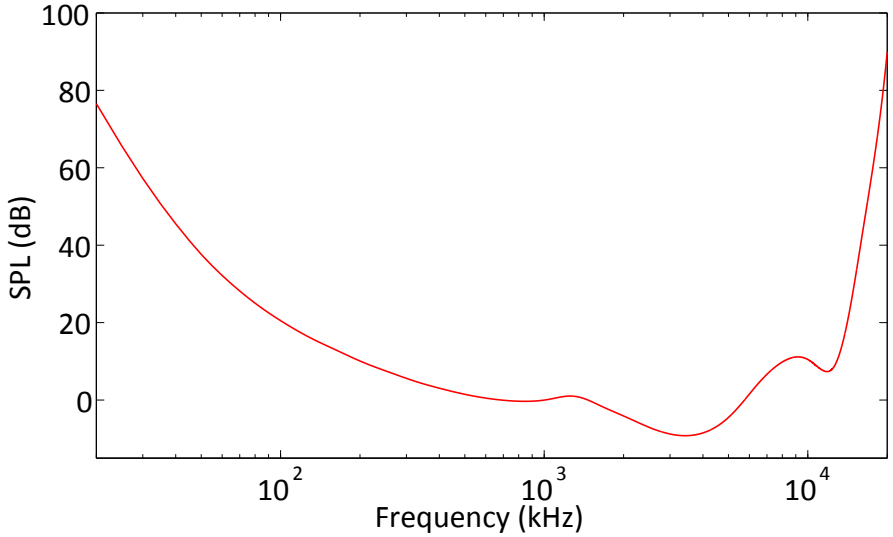


Figure 3.9: **Extended ISO-226 Frequency Weighting**

the power-law approximation, $\varepsilon = 1.5\,(p_{dB})^{-0.9}$ (shown in Figure 3.10). The effect of perceptual weighting on MLS compression is shown in Figure 3.11. Graphs of relative pressure estimates (3.9) and the resulting allocated per-mode error levels $\varepsilon^{j}_{\text{goal}}$ are shown in Figure 3.12 for several objects.

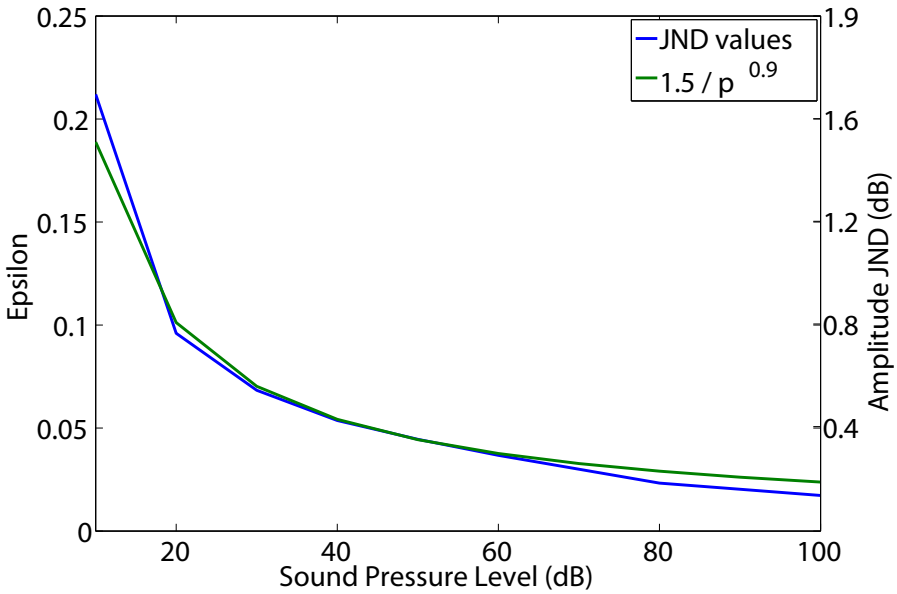


Figure 3.10: **Amplitude Just Noticeable Differences** are given for a 1 kHz tone at different amplitudes, along with the power-law approximation we use.

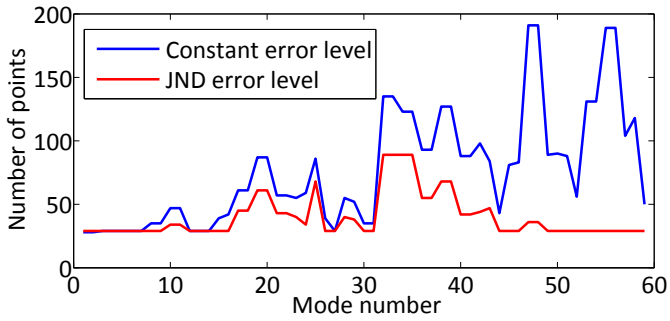


Figure 3.11: **Benefits of perceptual weighting:** Perceptually based error allocation allows fewer points to be used for quieter modes, compared to setting a constant $\varepsilon_{\text{goal}} = 0.037$ (0.32 dB)—the smallest error our model will assign. (Results are for the Dinner Plate.)

3.7 Results

Model	Geometry			MLS				Transfer		Symmetry	
	vertices	M	D (cm)	Uncompressed size (kB)	Adaptive size (kB)	Optimized size (kB)	Optimized R	Multipole size (kB)	Sym Multipole size (kB)	Symmetry-only R	# of congruent pairs
Large Bowl	70276	1194	200	983311	14835	7366	133 : 1	215715	163762	4.2 : 1	286
Backhoe Bucket	99253	494	96	574582	19734	9989	57 : 1	16353	16353	1.1 : 1	0
Heptoroid	81884	194	30	186158	6497	3064	60 : 1	470	470	1.0 : 1	0
Bell	41974	134	30	65912	408	200	329 : 1	2026	1668	1.6 : 1	23
Lego (huge)	7706	121	10	10927	319	183	59 : 1	111	88	3.2 : 1	26
Dinner Plate	50214	59	23	34718	74	39	897 : 1	182	135	4.6 : 1	17
Wine Glass	51434	46	19	27726	41	26	1061 : 1	108	69	5.7 : 1	16
Number 8	7843	37	30	3401	90	38	89 : 1	199	199	1.6 : 1	0
Letter A	7534	34	30	3002	88	42	71 : 1	167	167	2.0 : 1	0
Ellipsoid DB	6916	3037	40	251488	3602	2515	100 : 1	30463	30463	5.1 : 1	0
Rocks	3057	2727	26	101338	8444	3720	27 : 1	5606	5606	1 : 1	0

Table 3.1: **Compression Statistics** including compression ratios (R), transfer sizes, symmetry information, and the diameter (D) are given for various models. For transfer sizes, we report the total size and the size if only one of each congruent pair is stored. For the ellipsoid database of 71 ellipsoids, and the collection of 38 rock models, we report the total sizes and compression ratios, as well as the average number of vertices per model.

We present compression statistics in Table 3.1 for numerous models. The most complex nonsymmetric and symmetric models are the Heptoroid (see Figure 3.1) and the Large Bowl, respectively, and both cases observe excellent compression. For brevity, selected representative results are provided for collections of processed objects: plastic letters of the alphabet and plastic numerals. *Please see the accompanying video* for demonstrations, and comparisons of sounds generated using the uncompressed (before) and compressed (after) models. We provide comparisons for (i) before/after point-like impact sounds, and (ii) before/after sounds produced by various animations involving rigid-body contact

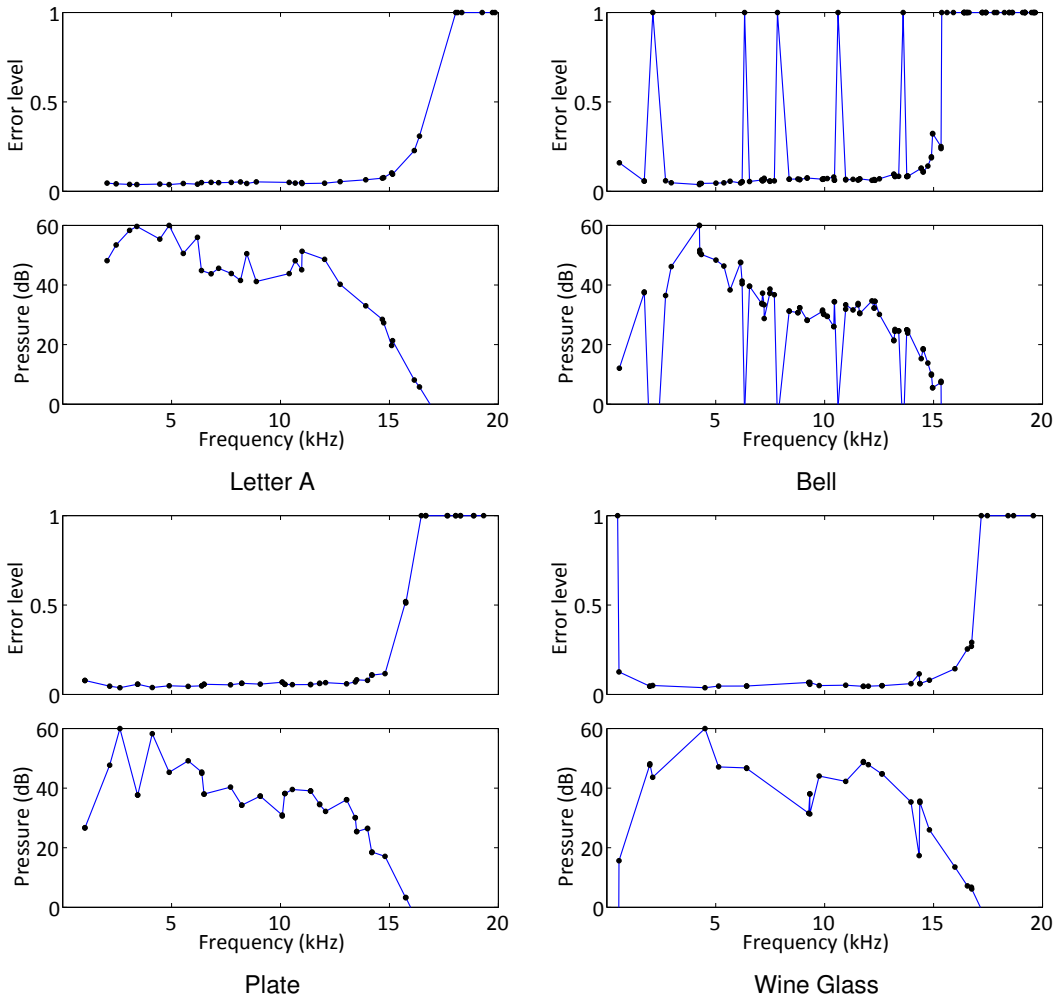


Figure 3.12: **Perceptually based errors** (Top) demonstrate that larger compression errors $\varepsilon^j_{\text{goal}}$ are used for modes which are either quieter or occur at frequencies of less perceptual importance; (Bottom) representative far-field modal pressures (normalized to 60dB) used in our error model, with ISO-226 weighting applied. Note that quieter modes are allocated larger MLS error goals, whereas the loudest modes are allocated the smallest errors. We clamp the maximum error to 1.

dynamics. In both cases, compression introduces very negligible (if any) audible sound differences, as desired.

For all models, the surface mesh we use is the surface of the tetrahedral mesh used for the eigenmode analysis. The tetrahedral mesh was chosen to conform to the original geometry well. The number of surface vertices directly

determines the uncompressed size of the eigenmode matrix, and while coarser meshes could be used, they will give coarser approximations of the eigenmodes.

Implementation details: Modal analysis and acoustic transfer computations are done using the implementation described in [171]. Linear tetrahedral meshes are generated using the Isosurface Stuffing algorithm [88]. We use Arpack to solve the generalized eigenvalue problem. Rigid-body dynamics are simulated using a solver based on [64]. All timing statistics are run on a machine with four Intel Xeon 7560 processors at 2.27GHz.

Ellipsoidal proxy soundbank: We compressed ellipsoid sound models from the ellipsoidal soundbank of [171]. Each ellipsoid was processed at the maximum size it could be used at (we chose to scale each ellipsoid so the maximum bounding box dimension was 40cm). For each ellipsoid mode with index m, the final compression level was calculated as $\tilde{\varepsilon}^m_{\text{goal}} = \min_{j \geq m} \varepsilon^j_{\text{goal}}$. This thresholding ensures that whenever the ellipsoid proxies are scaled down, the error level for each mode never increases. Despite these conservative error levels, by exploiting symmetry, we achieved a high compression ratio (100×) without introducing perceptible error (see Table 3.1). Surprisingly, at these compression rates, all compressed eigenmodes of the soundbank consume nearly as little memory as a single uncompressed model, greatly improving practical uses.

Symmetry: Applying intra- and inter-mode symmetry-based compression alone we observed up to 5.7× compression (see Table 3.1). Models with a large number of inter-mode congruent pairs, such as cylindrical models, e.g., 286/1194 modes (24%) for Large Bowl, were a particularly effective source of compression.

Comparison (Varying MLS polynomial degree, m): While we tend to use cubic $m = 3$ degree MLS approximations due to their good compression, lower degrees

are also possible when faster reconstruction is desired. A comparison of compression and reconstruction costs for varying degree, m, is shown in Table 3.2.

	m	Opt. size (kB)	Adaptive size (kB)	Opt. time (m)	Adaptive time (s)	Avg. vertex MLS time (μs)
Wine Glass	1	37	71	40	56	2.8
	2	27	48	23	31	7.8
	3	26	43	45	91	33
Lego (huge)	1	416	493	153	139	3.7
	2	226	383	308	87	8.8
	3	183	327	388	96	34

Table 3.2: **Dependence of compression and reconstruction costs on polynomial degree, m.**

3.8 Conclusion

We have presented a method for significantly compressing eigenmode matrices used in 3D modal sound synthesis using moving least squares approximations, and by exploiting symmetry. A perceptually based error model is proposed which allows the use of larger errors for modes at perceptually quieter frequencies. Overall, the method is able to achieve high compression rates, at the cost of a small runtime evaluation cost at sparse contact points.

Limitations and Future Work: While compressed modes result in very small differences in the rendered sounds, uncompressed modes should be used for the best quality. Building complex 3D modal sound models is expensive, and compression further increases preprocess times. Faster compression schemes (with better than "adaptive MLS" performance) would be useful. Preprocesses which directly estimate compressed representations would be helpful. Some applications may wish to use greater compression, which can be done at the cost of introducing noticeable sound differences. Runtime evaluation costs are reduced for lower MLS degree, m, which may be better for real-time applications. Com-

pression factors are larger (smaller) if higher (lower) resolution meshes are used. Our perceptual weighting model is based on radiative power, which requires acoustic transfer models for best results. Our approximation model is based on average-case (relative ℓ_2) error of the sound input-output model, however large pointwise relative errors can still occur depending on the specific location of the contact (input) and the listener (output). Also, frequency-dependent hearing sensitivities vary between listeners, and with amplitude, and users may or may not be able to hear differences. Future perceptual models might exploit inter-mode masking effects, which can be significant. Symmetry handling is approximate, and future works could better exploit it when large perceptual errors are admissible, and when approximate symmetries exist. Mode compression can result in acoustic transfer data dominating memory footprints for large objects, and future work should address the need for perceptually based transfer models.

CHAPTER 4

TOWARD ANIMATING WATER WITH COMPLEX ACOUSTIC BUBBLES

Frequency Deviation Vibration Magnitude

Figure 4.1: **Complex acoustic bubbles:** Our system is able to capture complex frequency effects due to bubbles' shapes and positions. (Left) Bubbles are colored blue/red if they are lower/higher than the theoretical Minnaert frequency for spherical bubbles, and depicts pitch rise near the surface. (Right) Bubbles are colored (blue/red) based on their (small/large) vibration magnitude.

In this chapter, we explore methods for synthesizing physics-based bubble sounds directly from two-phase incompressible simulations of bubbly water flows. By tracking fluid-air interface geometry, we identify bubble geometry and topological changes due to splitting, merging and popping. A novel capacitance-based method is proposed that can estimate volume-mode bubble frequency changes due to bubble size, shape, and proximity to solid and air interfaces. Our acoustic transfer model is able to capture cavity resonance effects due to near-field geometry, and we also propose a fast precomputed bubble-plane model for cheap transfer evaluation. In addition, we consider a bubble forcing model that better accounts for bubble entrainment, splitting, and merging events, as well as a Helmholtz resonator model for bubble popping sounds. To overcome frequency bandwidth limitations associated with coarse resolution fluid grids, we simulate micro-bubbles in the audio domain using a power-law model of bubble populations. Finally, we present several detailed examples of audiovisual

water simulations and physical experiments to validate our frequency model.

4.1 Introduction

Liquids, and the sounds they make, are pervasive in our daily lives. Whether it be a dripping faucet, a babbling brook, or your last glass of water, you have most likely heard sounds generated by fluids recently. While there has been significant work on understanding how fluids generate sound using bubbles, and breakthroughs in the *visual* simulation of water, there are no existing methods for computing a realistic *audiovisual* simulation of water with quality anywhere comparable to the purely visual component. For instance, current fluid sound approaches do not even simulate acoustic bubbles realistically by modeling their evolution using two-phase liquid simulations, but instead rely on single-phase flow solvers with *ad hoc* point-like bubble generation techniques [170, 110]. Such approximations are naturally much cheaper to compute, but, unfortunately, they have limited predictive value and ultimately limited realism. In contrast, we seek to understand whether one can simulate bubbly flows with sound from first physical principles, and what modeling challenges and trade-offs must be addressed.

In this paper, we explore a family of methods for sonifying detailed two-phase liquid animations such as the one shown in Figure 4.1. Given the complexity of liquid sound generation processes, there are many details we consider (see Figure 4.2 for an overview of our approach). Our approach begins with detailed multi-scale simulation of two-phase incompressible flow to resolve fine bubble geometry needed for higher frequency sounds[1]. Accurate modeling of

[1]Recall that Minnaert's frequency model predicts $f \approx 6.52/d_{mm}\ kHz$ (at STP) where d_{mm} is the

Figure 4.2: **Overview of our system:** From the fluid simulator, our method requires fluid surface geometry at each timestep, as well as bubble correspondences between timesteps. With this geometry, we compute each bubble's frequency and acoustic transfer magnitude, which are used to synthesize the bubble's sound.

surface tension is needed to resolve bubble pinch-off and subsequent topology changes. Individual bubble geometry is estimated and tracked, and we resolve bubble entrainment, merging, splitting and popping processes at sub-*ms* time scales, and sub-*mm* length scales.

Since the fluid flow is treated as incompressible for performance reasons, all acoustic fluctuations of the bubble/fluid/air system are handled using reduced-order vibration models. Toward this end, we propose a method for detailed bubble frequency analysis based on a "bubble capacitance" interpretation. Bubble pitch changes are perceptually important, but previous methods just model them using a parametric "chirp" as the bubble approaches the surface. In contrast, our method can resolve complex pitch shifts due to nonspherical bubble shapes, as well as nearby solid and air interfaces that can result in dramatic pitch decreases and increases, respectively (see Figure 4.1). We estimate the bubble frequency using a boundary element method, and since we analyze many bubbles, we propose a fast amortized matrix solver which effectively exploits inter-bubble similarities between matrix form factors and dense matrix solves.

To estimate a bubble's sound pressure at a listener position, we must faithfully estimate the surface-to-air acoustic transfer. To do so, we perform a stan-

bubble diameter in mm.

dard frequency-domain boundary element analysis, with surface vibration data input provided by our bubble-frequency solver. Unlike previous methods, this accounts for the near-field scene geometry to capture container resonance effects, such as the characteristic rising pitch of a container being filled up with water. For cheap, low-accuracy previews, we also propose a greatly simplified bubble-plane transfer model, based on precomputing a lookup table indexed by the size and depth of proxy bubbles.

During the final sound synthesis phase, we simulate the bubble oscillators using the estimated time-varying frequency. Bubble forcing models are devised to account for bubble excitations during surface entrainment, but also subsequent splitting and merging events. While prior works considered only Laplace pressure-jump forcing at entrainment, we leverage improved models of detailed surface-tension modeling (recently proposed by Deane and Czerski [49, 45, 46, 44]). To model bubble popping sounds, we describe a Helmholtz resonator model.

Since the fluid simulator's spatial resolution restricts bubble sizes, our synthesized sounds are inherently frequency band-limited. Therefore we propose a bandwidth extension scheme that performs audio-domain simulation of micro-bubbles based on power-law models of micro-bubble populations in breaking waves.

Finally, our results include multiple examples of dripping, pouring, and splashing phenomena, as well as results from laboratory experiments to validate our capacitance-based frequency model.

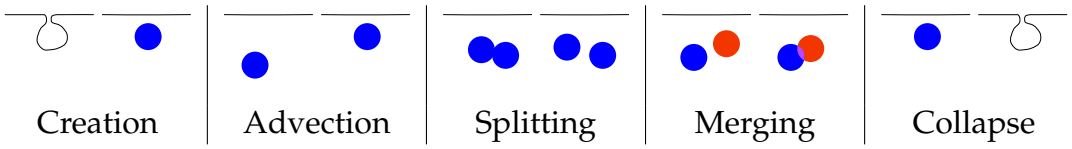


Figure 4.3: **Bubble Tracking:** We use colors to denote different bubble ids b_i. There are five types of bubble actions between timesteps. The left figure in each column denotes timestep i, and the right figure denotes timestep $i+1$. Creation or entrainment: no b_i overlap with b_{i+1}. Advection: b_{i+1} overlaps with one value of b_i. Splitting: one bubble id b_i overlaps with ≥ 2 different bubble ids, b_{i+1}. Merging: two different b_i's overlap with one b_{i+1}. Collapse: no b_{i+1} overlaps with a b_i.

4.2 Related Work

Fluid sound can be generated from a variety of sources, including vortex based sounds [72], fluid-structure interaction [71], shock waves, bubble popping [48], drop impact [73], and, the focus of this paper, harmonic bubble vibrations. Investigation of sounds produced by bubbles dates back almost a century, to the work of Lord Rayleigh [131] and William Bragg [27]. Minnaert calculated the frequency of isolated, spherical, harmonically vibrating bubbles [107]. Strasberg [149] described how a bubble's shape and the surrounding geometry affect its frequency, and by connecting frequency with capacitance went on to compute frequencies for sphere-plane and ellipsoid bubble geometries. Spratt et. al. [146] verified this model for simple nonspherical bubble geometries against a full acoustic scattering computation. We further extend this approach to support frequency computations with general nonspherical bubble systems and arbitrary air and solid interfaces.

Acoustic bubbles have been studied intensively, due to the impact bubbles have on physical processes, and the necessity of passive acoustic sensing of processes which are difficult to observe visually–Leighton's monumental work [92] provides a definitive summary. Acoustic bubbles represent a significant por-

tion of ambient ocean noise [124, 125], and contribute to gas exchange between the ocean and the air and affect ocean albedo [51], which in turn affects climate [24, 92]. They are important for "accurate quantification of a number of dynamic processes at the air-sea boundary, such as wave energy dissipation, gas exhange rates and the nature of rain" [84, 98, 126], such as using passive acoustic remote sensing [112, 52, 105, 166] and improvements (hopefully) in predictive computational models. Acoustic bubbles are even used by humpback whales during bubble net hunting [93]!

Fluid simulation has been a success story of computational physics, with widespread application in computer graphics and animation. However, most fluid simulators in graphics have focussed on single-phase free-surface flow [147, 54, 116, 28], and have only more recently tackled two-phase flow simulation [70, 102, 26, 6]. These works almost exclusively aim for visually plausible simulations with large timesteps, and not for acoustic bubble computations. Because of their visual richness, a variety of methods have been developed for simulating air bubbles [69, 70], and works range from tiny bubbles [30] to large volumes of bubbles [173], and complex thin-film interfaces [47]. These methods lack the ability to track individual acoustic bubbles while preserving their volumes accurately—for example, Kim et al. [79] artificially inflated bubbles to compensate for their volume change. However, bubble volume contributes significantly toward their estimated frequency, as demonstrated in our work. Consequently, we have built upon *Gerris* [120, 121], a finite-volume-based multigrid solver used in computational fluid dynamics, to more accurately simulate two-phase flows and track bubbles.

Fluid sound simulation: Despite the importance of bubble sounds, relatively little work has been done on simulating them. Van den Doel [163] proposed a sta-

tistical method to generate bubble sounds. More recently, two works [170, 110] have proposed more physically based methods. However, due to the sheer computational difficulty of predictive modeling of bubble entrainment processes, these methods have relied on single-phase liquid simulators with *ad hoc* stochastic models to estimate point-like bubble creation rates and size distributions, with the unfortunate consequence that bubble creation rates are either unrealistic [110] or must be laboriously hand-tuned for examples [170]. Furthermore, these single-phase flows solvers can not estimate realistic nonspherical bubbles, and they lack realistic time-varying bubble frequencies. Additionally, Zheng and James [170] use an acoustic transfer approximation which cannot capture scattering effects from enclosing solid interface, while Moss et al. [110] ignore acoustic radiation entirely.

Multi-frequency vibration: For small acoustic bubbles, it is usually assumed that they vibrate at a single frequency. However, it is possible that other vibration modes can radiate sound at different frequencies. Lamb [89] investigated higher-order vibration modes using a first-order perturbation analysis, and concluded that they do not radiate efficiently enough to be important. A series of papers by Longuet-Higgins [99, 100, 101] showed that in certain situations, resonant coupling can occur between shape modes and the volume mode, and higher vibration modes can radiate as a monopole, causing a single bubble to produce sound at several frequencies. The recent work in graphics by [110] proposed using a related multi-frequency zonal-harmonic vibration model. While multi-frequency radiation has been observed in certain specific experimental conditions [37], it is believed that under real-world forcing conditions these coupling effects are less important [101], and arguments have been given that shape-mode coupling accounts for a very small portion of the total air-domain sound radia-

tion of bubbles [98]. Longuet-Higgins himself also mentions that his model is not valid for realistic pinch-off scenarios, which would require fully nonlinear equations [100].

Experimental evidence to support multi-frequency bubble sounds is also lacking. Medwin and Beaky [106] analyzed a large number of cases of single- and multi-bubble events (over 2000) generated in a wave tank. They laboriously classified four types of bubble radiation. The large majority of events show no signs of multiple frequencies. The one type of event ("type D") that does show multiple frequencies is speculated to be caused by interference from two bubbles. Therefore, while it is possible for a single bubble to radiate at multiple frequencies, it seems to be rare, and we ignore it in the present study.

4.3 Fluid Simulation

Due to the dependence of a bubble's frequency on its size, shape, and position, accurate fluid simulation is necessary for realistic sound. We solve the incompressible, variable-density Navier-Stokes equations with surface tension.

$$\rho\left(\partial_t \mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \nabla\cdot(2\mu\mathbf{D}) + \sigma\kappa\delta_s\mathbf{n} \tag{4.1}$$

$$\partial_t\rho + \nabla\cdot(\rho\mathbf{u}) = 0 \tag{4.2}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{4.3}$$

with $\mathbf{u}$ the fluid velocity, ρ the fluid density, μ the dynamic viscosity, and $\mathbf{D}$ the deformation tensor defined as $D_{ij} = (\partial_i u_j + \partial_j u_i)/2$. The surface tension force is only nonzero on the interface, which the Dirac distribution δ_s signifies. σ is the surface tension coefficient, κ is the surface curvature (not related to our κ in equation (4.4)), and $\mathbf{n}$ is the surface normal.

For two-phase flows, densities ρ_1, ρ_2 and viscosities μ_1, μ_2 are given for the first and second fluids respectively. The continuous volume fraction field $c(\mathbf{x}, t)$ is used to denote the volume of the first fluid and defines the density and viscosity as

$$\rho(c) = c\rho_1 + (1 - c)\rho_2$$

$$\mu(c) = c\mu_1 + (1 - c)\mu_2$$

The advection equation for the density (4.2) is then replaced with an equivalent equation for the volume fraction

$$\partial_t c + \nabla \cdot (c\mathbf{u}) = 0.$$

We use the open-source *Gerris* solver [120, 121], which is based on a finite volume method implementation with an adaptive octree data structure, parallel multigrid Poisson solver, and an accurate surface tension model. *Gerris* enables high-fidelity simulation of bubble shapes and motion, with good volume preservation properties, as well as realistic bubble formation from surface entrainment. However, despite its merits, there are significant spatial and temporal resolution requirements in order to resolve bubble entrainment and subsequent topological events, and thin interfaces. In our simulations, we use spatial resolutions on the order of [0.625mm, 5mm], timesteps (determined by CFL conditions) on the order of $[30\mu s, 250\mu s]$, and spatial adaptation near interfaces. Nevertheless, our ability to resolve tiny bubbles below 1 *mm* is limited. Also, *Gerris* has no way to preserve thin films when bubbles come into contact with each other or the air surface, so merging events are over estimated and surface popping is immediate.

4.4 Bubble Identification and Tracking

The link between the fluid simulation and our sound pipeline is the interface geometry, consisting of the surface of each bubble and the enclosing fluid volume. We describe here how this is implemented in *Gerris*, but note that any fluid simulation method which can provide this geometry will be compatible with our method.

Our representation of the interface geometry is given by c on the octree grid. Specifically, bubbles are connected components of cells where $c < 1$, which are identified and uniquely numbered with a flood fill algorithm. To track bubbles between time steps, a new variable b_i is defined at timestep i and initialized to the number of each bubble. The b_i variable is advected between time steps. Overlaps between the current b_{i+1} and the previous b_i advected from the last timestep are used to correlate the bubbles between time steps. There are several situations:

1. **New bubble:** No b_i touching b_{i+1}
2. **Advected bubble:** One value of b_i touching one value of b_{i+1}
3. **Split bubble:** Two or more values of b_{i+1} overlap one of b_i
4. **Merged bubbles:** Two or more values of b_i overlap one of b_{i+1}
5. **Collapsed bubble:** No b_{i+1} touching a value of b_i

These are illustrated in Figure 4.3. The CFL condition ensures that interface fragments do not move more than one grid cell per timestep, and we did not observe any bubbles getting lost during tracking. Marching cubes [94] is used to generate interface geometry for each bubble, and for the enclosing fluid volume.

4.5 Bubble Frequency Estimation

After a brief introduction to bubble vibrations (§4.5.1), we describe a new model for estimating the instantaneous bubble frequency (§4.5.2, §4.5.3), and an efficient algorithm that amortizes computation across many bubbles (§4.5.4).

4.5.1 Bubble Basics

The equations of spherical bubble vibration were originally proposed by Minnaert [107] and are described in detail in Chapter 3 of [92]. Briefly, when bubbles are formed, they vibrate, creating pressure waves which travel through the fluid and then pass through the fluid-air interface into the air. The simple harmonic oscillator model is similar to a linear spring-damper-mass system, where the spring forces are provided by internal gas pressure and surface tension, and the mass is due to the surrounding liquid. We use the "volume-pressure" frame, so that the infinitesimal volume pulsation of a bubble $v = V(t) - V_0$ (where V_0 is the average volume) satisfies

$$m\ddot{v} + \alpha\dot{v} + \kappa v = p(t) \tag{4.4}$$

where $p(t)$ is a forcing term in units of pressure. The equation is usually divided through by m and written as

$$\ddot{v} + 2\beta\dot{v} + \omega^2 v = \frac{p(t)}{m}$$

with $\omega^2 = \frac{\kappa}{m}$. For a bubble of equivalent radius r, $\beta = \omega\delta/\sqrt{\delta^2 + 4}$, with $\delta = \delta(\omega, r) = \delta_{rad} + \delta_{vis} + \delta_{th}$ where

$$\delta_{rad} = \frac{\omega r}{c} \qquad \delta_{vis} = \frac{4\mu}{\rho\omega r^2} \qquad \delta_{th} = 2\frac{\sqrt{\psi - 3} - \frac{3\gamma-1}{3(\gamma-1)}}{\psi - 4}$$

with $\psi = \frac{16}{9(\gamma-1)^2}\frac{G_{th}g}{\omega}$, c is the speed of sound in the fluid, μ is the liquid's shear viscosity, ρ is the fluid density, γ is the gas' heat capacity ratio, $G_{th} = \frac{3\gamma p_f}{4\pi\rho D_g}$ is the thermal damping constant at resonance, D_g is the gas' thermal diffusivity, g is gravitational acceleration, and p_f is the hydrostatic pressure of the liquid. A full derivation is provided by Leighton [92].

Minnaert computed ω for a spherical bubble. However, nearby fluid-air and fluid-solid surfaces, as well as bubble shape, can affect ω, and are the reason for the familiar pitch shift as bubbles approach the fluid surface. In the following sections, we describe one of our main contributions: a method for accurately computing ω (by computing κ and m) for complex geometries, and thereby providing more realistic sound.

4.5.2 Frequency Model

Extending Strasberg's [149] derivation, we seek to better estimate the bubble frequency ω, by more accurately modeling the scene-specific effective stiffness, κ, and effective mass, m. We consider a bubble's infinitesmal volume pulsation $v(t)$ in the volume-pressure frame (4.4), in the undamped case ($\alpha = 0$).

The effective stiffness (κ) accounts for internal gas and surface tension effects, and is the simplest of the two values. The stiffness in the volume-pressure frame is the rate of change of bubble pressure with volume,

$$\kappa = -\frac{dp}{dv}. \tag{4.5}$$

Assuming a polytropic gas law, the reference volume V_0 and pressure P_0 are related to the modified volume $V = V_0 + v$ and pressure $P = P_0 + p$ by

$$P_0 V_0^\gamma = PV^\gamma,$$

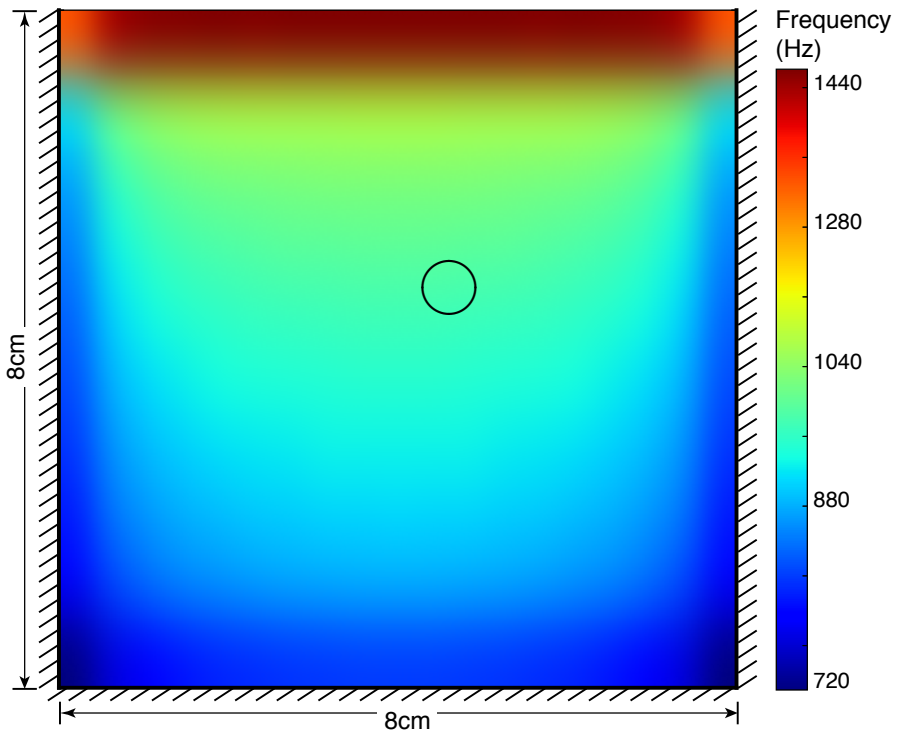


Figure 4.4: **Spatially varying bubble frequency** (in Hz) depicted for a spherical bubble of 3mm radius (Minnaert frequency of approximately 1100 Hz). The pitch lowers as the bubble nears rigid walls (left, right, and bottom), and rises sharply as the bubble nears the fluid surface (top). Even in this small 8cm-by-8cm tank, the bubble's frequency can differ by over 700 Hz depending on position. Note that for this figure, frequencies were only sampled 3.6mm from the boundary (wall or surface), and extrapolated.

where γ is the polytropic index; we use $\gamma = 1.4$ for air. Evaluating the derivative (4.5) we obtain the bubble stiffness,

$$\kappa = \frac{\gamma P_0}{V_0}. \tag{4.6}$$

The bubble air pressure P_0 is the sum of the hydrostatic pressure (approximately $P_{\text{atm}} + \rho g d$ for a bubble at depth d), and surface tension ($\frac{2\sigma}{R_0}$ for a spherical bubble). To support pressure and surface tension for nonspherical bubbles in complex fluid domains, we obtain a reliable approximation of P_0 from our two-phase fluid simulator. Observe that κ only depends on the internal gas and surface tension, which is consistent with Minnaert's derivation [92]. Interestingly, note

that κ is incapable of producing increasing pitch as a bubble rises (it actually predicts the opposite dependence on d), and thus the mass factor is responsible for this effect.

The effective mass (m) can be derived similar to Strasberg [149], but with consideration for complex fluid domains. The key idea is to equate the oscillator kinetic energy $\frac{1}{2}m\dot{v}^2$, for a given harmonic bubble volume velocity $\dot{v}$, with the kinetic-energy volume integral of the surrounding fluid, W_f, to determine m. W_f is also the work that the bubble does to the fluid. In order to compute the fluid kinetic energy, we exploit the fact that bubbles are acoustically compact low-frequency sources [2], and thus the surrounding fluid's vibration is well approximated by incompressible, irrotational flow modeled using Laplace's equation. Specifically, let the surrounding fluid's particle velocity be $\nabla\phi$, where ϕ is the velocity potential which satisfies

$$\nabla^2\phi(x) = 0, \qquad x \in \Omega,$$

subject to a pressure-like Dirichlet boundary condition on the bubble, $\phi = \phi_b$, and other suitable boundary conditions elsewhere (discussed later). Therefore we can express the effective mass as

$$m = \frac{2W_f}{\dot{v}^2} = \frac{2}{\dot{v}^2}\left(\frac{\rho}{2}\int_\Omega (\nabla\phi)^2\, d\Omega\right).$$

Converting the volume integral to a boundary integral using Green's first identity we obtain

$$\int_\Omega (\nabla\phi)^2\, d\Omega + \underbrace{\int_\Omega \phi\nabla^2\phi\, d\Omega}_{\approx 0} = -\int_{\partial\Omega} \phi\, \partial_n\phi\, dS,$$

where the minus sign is due to normals pointing out of the fluid domain Ω (and into air or solid domains). Denoting the bubble, air, and rigid surfaces as Γ_b, Γ_a,

[2] Note that for a bubble at 1 atm pressure, we have $fR \approx 3m/s$, or $\omega R \approx 19m/s$, so that $kR = \omega r/c \approx 0.013 << 1$. Also note that $\lambda/R \approx 114$.

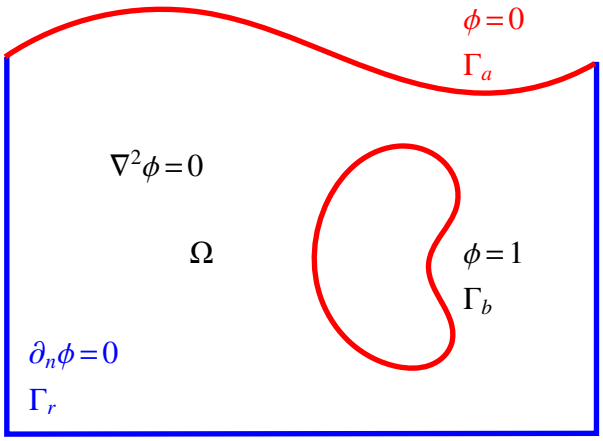


Figure 4.5: **Interior Laplace BVP for bubble capacitance:** We use the solution's $\partial_n\phi$ gradient on the bubble boundary Γ_b to compute the bubble capacitance using (4.9), and on the air boundary Γ_a to evaluate acoustic radiation (in §4.6).

and Γ_r, the mass integral becomes

$$\begin{aligned} m &= -\frac{\rho}{\dot{v}^2}\left(\int_{\Gamma_b} \phi\, \partial_n\phi\, dS + \int_{\Gamma_a} \phi\, \partial_n\phi\, dS + \int_{\Gamma_r} \phi\, \partial_n\phi\, dS\right) \\ &= m_b + m_a + m_r. \end{aligned}$$

The terms can be evaluated by making use of the boundary conditions for the generalized problem (see Figure 4.5). The air-interface m_a contribution is zero since $\phi = 0$ on Γ_a, as the acoustic pressure $p = -\rho\frac{\partial\phi}{\partial t} = -i\omega\rho\phi = 0$ there. The rigid-surface m_r contribution is also zero: the acoustic particle velocity must be zero in the normal direction on the rigid boundary, and thus $\partial_n\phi = 0$ on Γ_r. Therefore, *the only mass contribution arises from the bubble term* $m = m_b$.

4.5.3 Capacitance Interpretation of Bubble Frequency

Since the velocity potential is constant on the surface of the bubble, $\phi = \phi_b$, it can be taken outside the integral,

$$m = -\frac{\rho}{\dot{v}^2} \int_{\Gamma_b} \phi \, \partial_n \phi \, dS \approx -\frac{\rho}{\dot{v}^2} \phi_b \int_{\Gamma_b} \partial_n \phi \, dS.$$

Then since the volume velocity is $\dot{v} = -\int_{\Gamma_b} \partial_n \phi \, dS$, it follows that $m = \rho \phi_b / \dot{v}$. The bubble is a uniform-potential conductor-like surface, so Strasberg noticed that treating ϕ as electrostatic potential and $\dot{v}$ as flux, the ratio $\dot{v}/\phi$ is mathematically equivalent to 4π times the surface's electrostatic capacitance, C. Therefore, the effective mass can be identified as

$$m = \frac{\rho}{4\pi C}. \tag{4.7}$$

By combining equations (4.6) and (4.7), the bubble frequency can be given in terms of its capacitance,

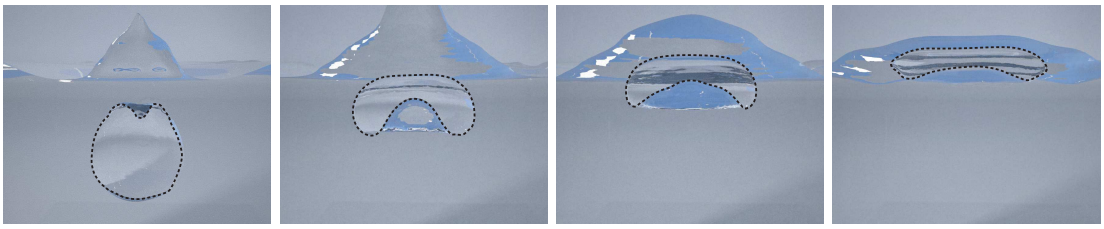
$$\boxed{\omega^2 = \frac{\kappa}{m} = \frac{4\pi\gamma P_0}{\rho V_0} C.} \tag{4.8}$$

Bubble capacitance BVP: To compute the capacitance we can interpret the bubble as a conductor with a unit potential boundary condition ($\phi_b = 1$), the fluid-air surface as a conductor at zero potential ($\phi_a = 0$), and rigid interfaces as insulators ($\partial_n \phi_r = 0$). Our generalized "bubble capacitor" boundary value problem (BVP) is shown in Figure 4.5.

Given the solution to Laplace's equation $\hat{\phi}$ for this BVP, we compute the capacitance as

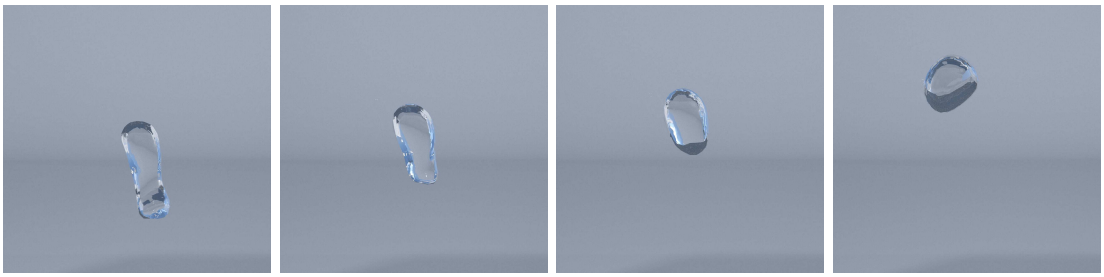
$$\boxed{C = -\frac{1}{4\pi} \int_{\Gamma_b} \partial_n \hat{\phi} \, dS.} \tag{4.9}$$

Since this formula only requires $\partial_n \hat{\phi}$ on Γ_b, we can solve for each bubble's capacitance using a boundary integral formulation of Laplace's equation. We discuss



(a) $\omega = 1$ $C = 1$ (b) $\omega = 1.14$ $C = 1.31$ (c) $\omega = 1.27$ $C = 1.61$ (d) $\omega = 1.72$ $C = 2.96$

Figure 4.6: **Capacitance-based frequency estimation for a rising bubble:** We recover increasing "chirp-like" frequency and capacitance (normalized) as the bubble (initial radius $R = 5.8$ mm) nears the surface (a-d). The rising pitch produced as the bubble's water layer (lamella) thins corresponds to a thin-plate capacitor of increasing thinness.



(a) $\omega = 1$ $t = 0$ (b) $\omega = 0.95$ $t = 3ms$ (c) $\omega = 0.91$ $t = 6ms$ (d) $\omega = 0.85$ $t = 13ms$

Figure 4.7: **Shape-dependent bubble frequency** is demonstrated here for a simulated bubble undergoing natural shape changes. Frequency is normalized. Our bubble frequency estimation method can resolve musical pitch fluctuations occuring on semi-tone magnitudes, on the order of 10ms.

an optimized bubble capacitance solver in §4.5.4. Finally, our approach reproduces and generalizes the frequency models of Minnaert and Strasberg, and supports nontrivial frequency estimation in complex fluid geometries (see Figure 4.6), and for complex bubble shape changes (see Figure 4.7).

4.5.4 Fast Amortized Solution of Capacitance BVP

We now describe an efficient method for estimating bubble capacitances that exploits common computations between the bubbles for speed. We can use

the boundary element method (BEM) to solve the capacitance BVP using established codes for the interior Laplace problem with mixed boundary conditions (BCs) [9].

BEM BVP matrix structure: After discretizing the direct boundary integral equation formulation of the interior Laplace problem for the velocity potential associated with a single bubble, we arrive at the linear matrix problem,

$$\mathbf{H}\boldsymbol{\phi} = \mathbf{G}\boldsymbol{v},$$

with the following block structure,

$$\begin{bmatrix} H_{bb} & H_{ba} & H_{bs} \\ H_{ab} & H_{aa} & H_{as} \\ H_{sb} & H_{sa} & H_{ss} \end{bmatrix} \begin{pmatrix} \phi_b \\ \phi_a \\ \phi_s \end{pmatrix} = \begin{bmatrix} G_{bb} & G_{ba} & G_{bs} \\ G_{ab} & G_{aa} & G_{as} \\ G_{sb} & G_{sa} & G_{ss} \end{bmatrix} \begin{pmatrix} v_b \\ v_a \\ v_s \end{pmatrix},$$

where $\boldsymbol{\phi}$ is the BEM vector of potential values and $\boldsymbol{v}$ is the BEM vector of (outward) normal derivative values, $\frac{\partial \phi}{\partial n}$; here the three boundary regions are denoted by b (bubble), a (air), and s (solid). Applying the Capacitance BVP boundary conditions (see Figure 4.5), we arrive at the linear system $\mathbf{Ax} = \mathbf{b}$ with the block structure,

$$\underbrace{\left[\begin{array}{c|cc} G_{bb} & G_{ba} & H_{bs} \\ \hline G_{ab} & G_{aa} & H_{as} \\ G_{sb} & G_{sa} & H_{ss} \end{array}\right]}_{\mathbf{A}} \underbrace{\begin{pmatrix} v_b \\ v_a \\ -\phi_s \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} H_{bb} \\ H_{ab} \\ H_{sb} \end{bmatrix} \left(\phi_b\right)}_{\mathbf{b}},$$

where the unknowns are $v_b \in R^{n_b}$, $v_a \in R^{n_a}$, and $\phi_s \in R^{n_s}$. Solving this system provides three quantities: v_b which helps approximate $\int_{\Gamma_b} \partial_n \hat{\phi}\, dS \approx a_b^T v_b$ and thus (4.9) for the bubble capacitance; v_a which describes the free surface vibration, and will be used for acoustic radiation modeling (in §4.6); and ϕ_s which is ignored since it is not needed by our application.

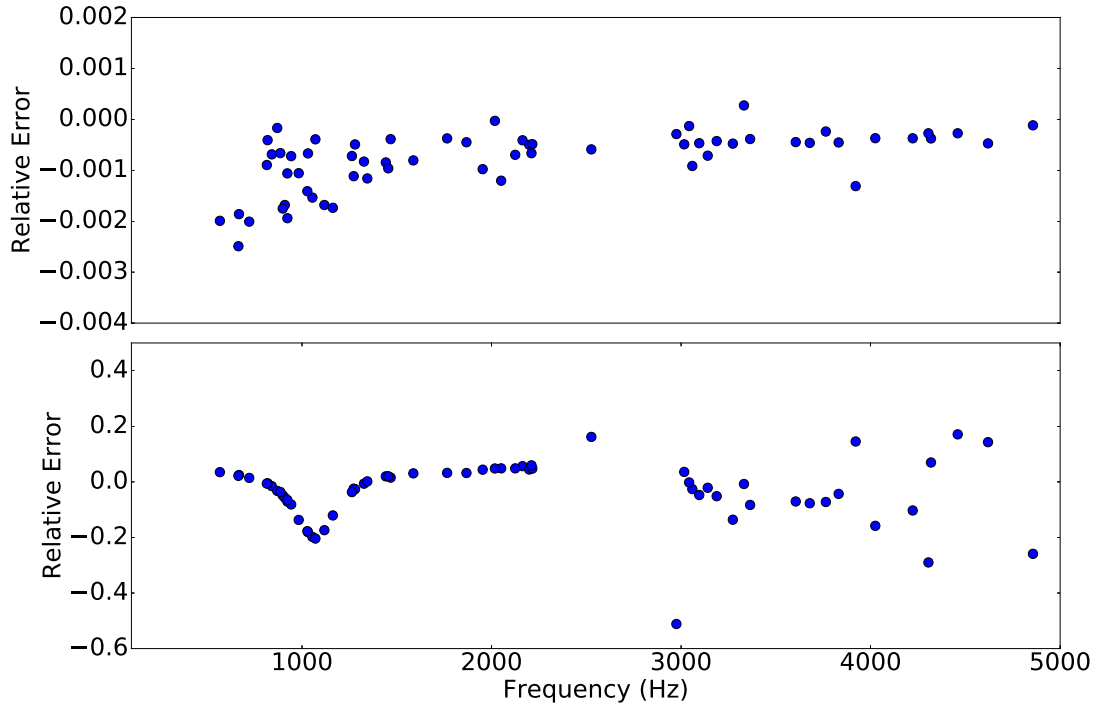


Figure 4.8: **Solver Error:** For one timestep of the pouring faucet example, we meshed the domain at a high resolution (5mm maximum edge length), and computed frequency and transfer for all the bubbles. Although we use aggressive mesh simplification, the relative errors for our frequency solver compared to the high resolution results are very small (top). Pressure magnitude errors (bottom) are tolerable in our range of interest, and increase at higher frequencies as expected.

In our implementation we use the BEM++ software library (version 2.0) [142] to evaluate the various **G** and **H** matrix blocks; we used Galerkin discretization (which is more robust to meshing irregularities than collocation schemes) with constant elements for Neumann data (on Γ_s), and piecewise linear elements for Dirichlet data (on Γ_b and Γ_a).

Fast amortized matrix solver: For a given frame, we solve the matrix problem many times for many bubbles. A naive LU-based evaluation of $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ would

require $O\left((n_b + n_a + n_s)^3\right)$ flops. If we denote the four blocks of $\mathbf{A}$ by

$$\mathbf{A} = \left[\begin{array}{c|c} G_{bb} & B \\ \hline C & D \end{array}\right],$$

we observe that the huge lower-right "domain" submatrix D relating the self-effect form factors for the air and solid boundaries ($\Gamma_a \cup \Gamma_s$) is constant across problems. We can exploit this fact for fast evaluation of $\mathbf{A}^{-1}\mathbf{b}$ for different bubbles. By exploiting the Sherman-Morrison-Woodbury formula for U [66], the matrix inverse can be written as

$$\mathbf{A}^{-1} = \left[\begin{array}{c|c} X & Y \\ \hline Z & U \end{array}\right]$$

where

$$X = \left(G_{bb} - BD^{-1}C\right)^{-1}, \tag{4.10}$$

$$Y = -XBD^{-1}, \tag{4.11}$$

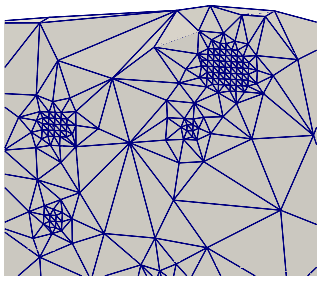
$$U = D^{-1}(I - CY), \text{ and} \tag{4.12}$$

$$Z = -UCG_{bb}^{-1}. \tag{4.13}$$

By carefully exploiting common subexpressions and cached LU factorization of the large D block, the product $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ can be evaluated in $O\left((n_a + n_s)^2\right)$ flops for small bubbles of size $n_b = O(1)$. Please see Appendix B.1 for explicit algorithmic details. In our examples, we observe that the amortized capacitance solver accelerates bubble frequency estimation by 10-12x for small bubbles, while for very large bubbles the speed-up may only be 3x, but still worthwhile.

4.5.5 Adaptive Meshing

The generation of meshes used for BEM analysis should be discussed briefly, since adaptive meshing is needed to keep our frequency solve times manageable.



Detailed meshes are obtained from the fluid simulator using Marching Cubes, which we adaptively simplify using Quadric Error Simplification [62]. However, we must properly resolve interfaces and inter-surface gaps as bubbles approach the fluid surface or container walls in order to compute accurate capacitance values. Therefore we use a sizing function that ensures each triangle's maximum edge length is less than its distance to the nearest bubble. The inset shows an illustrative example. Although we use aggressive mesh simplification, our errors (shown Figure 4.8) are still tolerable.

4.6 Bubble Acoustic Transfer

We now describe two acoustic transfer solvers that estimate the pressure amplitude at the listener's position(s) from an acoustic bubble vibrating with unit pressure: (1) a BEM-based solver that includes scene geometry (§4.6.1), and (2) a fast but very approximate solver that uses a proxy bubble-plane transfer model (§4.6.2). In practice, we sample frequency and bubble-to-ear transfer values at a fixed rate during the lifetime of the bubble; in our implementation these solves are done every 1ms, and interpolated with a cubic spline. These transfer values are used for sound synthesis later in §4.8.

4.6.1 BEM-based Acoustic Transfer Solver

We now describe how to approximate realistic sound amplitudes from an harmonically vibrating fluid surface. Since sound scattering and resonance effects

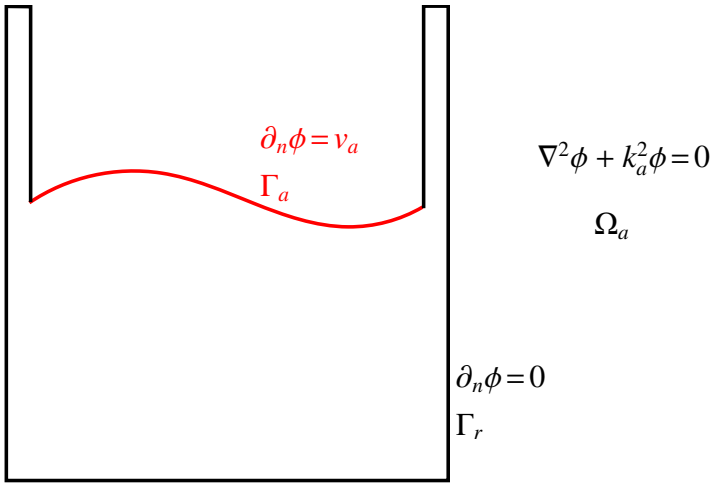


Figure 4.9: **Exterior Helmholtz BVP for acoustic radiation**

from external geometry (such as the walls of a glass container) can introduce perceptually important and pitch-dependent amplitude variations (such as when a glass is filled up with water) we seek to include near-field scene geometry in our transfer solver that has been neglected in prior works [170].

Mathematically we approximate solutions to the exterior wave radiation problem specified by the Helmholtz equation

$$(\nabla^2 + k^2)p(x) = 0, \qquad x \in \Omega_a,$$

where $k = \omega/c_a$, and c_a is the speed of sound in air. As boundary conditions, we impose a vibration BC on the fluid-air interface, and a no-vibration BC on the rigid scene geometry (see Figure 4.9). Following our bubble frequency solve, the vibration of the fluid-air surface is known[3] from $\partial_n\hat{\phi}$ on Γ_a, which in turn becomes input boundary data for the air-domain acoustic transfer estimation.

To support arbitrary scene geometry, we use established BEM solvers for the exterior Helmholtz radiation BVP. In our implementation, we use the BEM++

[3]We refer to the pressure gradient as a velocity, but they are merely proportional: $\partial_n p = -i\omega\rho V_n$ for a harmonic vibration of velocity V_n. Since the normal velocity is continuous across the fluid-air interface, $-i\omega V_n = \partial_n p_a/\rho_a = \partial_n p_f/\rho_f$, and so up to a multiplicative constant density factor we can say that $\partial_n p_a$ and $\partial_n p_f$ are equal velocity-like quantities.

implementation [142] A major practical task is generating a suitable scene mesh with BC data at each time step. We first mesh the fluid and external container together to generate detailed geometry for the external radiation problem (see Figure 4.11). Since these meshes are typically too detailed for efficient BEM analysis, we again use adaptive meshing (as in §4.5.5) to decimate the mesh. We further restrict the largest edge length so as to sufficiently resolve the smallest wavelength; in our examples, we have used a 3cm limit which can resolve $\lambda_a = 17.2cm$ at the upper 2kHz range we simulate. Finally the interior capacitance BVP and exterior transfer BVP may have different meshes for the fluid-air interface, and therefore we interpolate the previously obtained velocity-like solution data, v_a (i.e., $\partial_n \hat{\phi}$), from the interior mesh to the mesh of the exterior fluid-air interface Γ_a. The pressure BC on that interface is the same up to a scaling factor, since $\partial_n p_f = \frac{\rho_f}{\rho_a} \partial_n p_a = \frac{\rho_f}{\rho_a} \partial_n \hat{\phi}$.

4.6.2 Fast Bubble-Plane Proxy Transfer

Transfer computations can be expensive for detailed water surfaces and scenes. For many applications and "fast preview" renderings, we can use a cheap transfer model based on a simplified bubble-plane geometry (see Figure 4.10).

To do this, we precompute a lookup table of transfer values for spherical bubbles of various radii (r) at various depths (d) beneath a planar water surface. We sample bubble radii from 0.25mm to 1cm in 0.25mm increments, and depths down to 8cm in 0.5mm increments. For each radius and depth, we solve the interior frequency BVP and exterior radiation BVP. Then we fit

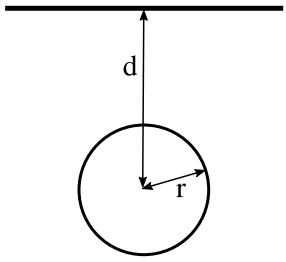


Figure 4.10: **Fast proxy transfer model**

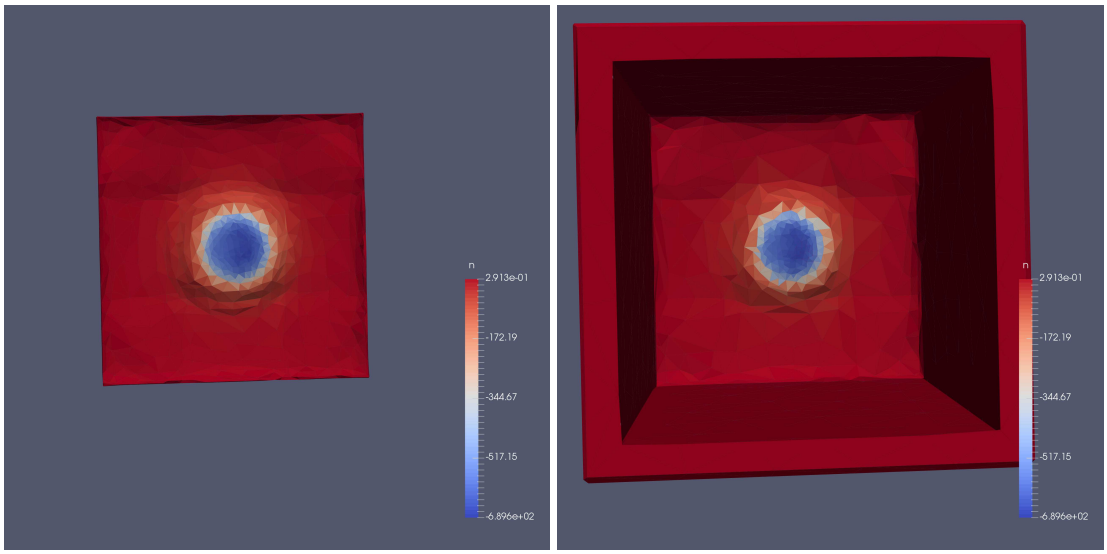


Figure 4.11: **Interpolating velocity BC data** for a rising bubble in a square container (top view): (Left) velocity BCs from the interior solve geometry (on Γ_a) are (Right) interpolated onto the mesh (of $\Gamma_a \cup \Gamma_r$) for the exterior Helmholtz BVP.

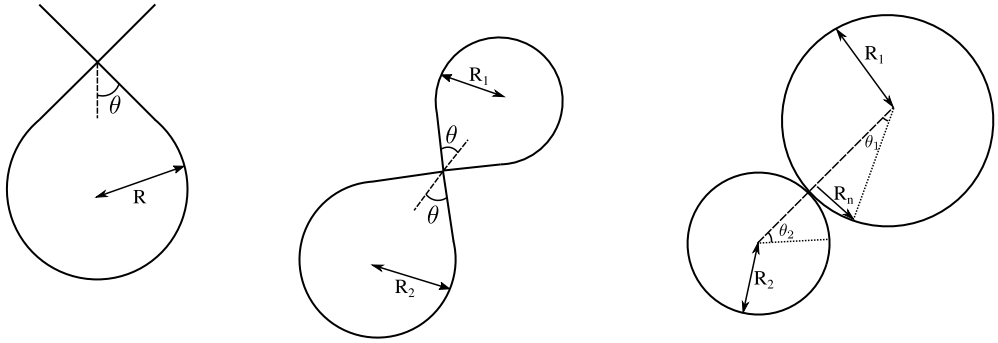
a multipole expansion [171] to the bubble's exterior radiation data, and only store the multipole coefficients (as opposed to storing all the boundary data). Further, we assume the bubble's response is cylindrically symmetrical, so we only store the zonal multipole coefficients.

At runtime, instead of solving the exterior BVP, we compute the equivalent spherical radius of the bubble, and the distance to the fluid surface. These values are used to lookup the closest set of multipole coefficients in our database, which are evaluated at the listening position. When performing lookups in our database, we first find the closest radius, then find the closest depth sample with that radius. It took 2.2 hours to construct the database using 32 cores. Runtime evaluation times for each example are given in Table 4.1.

4.7 Bubble Forcing

There are several mechanisms which can drive bubble vibrations. Previous fluid sound work in graphics has used Laplace pressure forcing, which approximates entrainment forcing by a pressure jump due to surface tension. The additional pressure, $p_\sigma = \frac{2\sigma}{R}$ (R is the mean bubble radius), provides an initial impulse to the bubble oscillator (possibly smoothed in time). However, estimates of the Laplace pressure jump, as well as hydrostatic pressure and shape mode coupling effects, show that they represent a minor ($< 10\%$) amount of the total forcing [127]. A recent set of papers from Deane and Czerski [49, 45, 46, 44] propose a family of models based on neck collapse (for entrainment and splitting events) and neck expansion (for merging events) where surface tension effects account for the majority of forcing (summarized in Figure 4.12).

Entrainment: The forcing of bubbles released from an underwater tube were analyzed in [49, 45]. As the bubbles separate from the nozzle, a conical neck forms with very sharp curvature (causing high surface tension) at the tip. At separation, surface tension causes the neck to rapidly shoot into the bubble,



(a) (b) (c)

Figure 4.12: **Bubble forcing:** There are 3 types of forcing events we model: (a) entrainment, (b) splitting, and (c) merging.

quickly decreasing the bubble's volume, which forces oscillations. A similar situation happens during bubble entrainment: a neck is formed as a bubble separates from the fluid surface. This geometry is illustrated in Figure 4.12a. Their time-dependent forcing model based on a conical neck is

$$f(R,t) = -\frac{9\kappa\sigma\eta p_{in,0}\sqrt{1+\eta^2}}{4\rho R^3}t^2, \tag{4.14}$$

where κ is the gas polytropic index, σ is the surface tension coefficient, $p_{in,0}$ is the equilibrium pressure in the bubble, R is the equilibrium radius, ρ is the fluid density, and η is the slope of the neck cone ($\eta = \tan(\theta)$). We have multiplied their original equation by ρR^2 to transform to the correct units (pressure). We sample the neck angle uniformly between $[20^o, 55^o]$, which corresponds to $\eta \in [0.36, 1.43]$. It was observed that this forcing function is valid for approximately 300-400 μs for 2mm bubbles (Minnaert frequency = 1.6 kHz, period = 625 μs). For a bubble with period τ, we use this forcing function for the $\min\left(\frac{1}{2}\tau, 600\mu s\right)$, and set it to 0 afterwards.

Splitting: For bubble splitting events, the same mechanism is proposed in [46] as the main source of forcing. Both child bubbles are driven by the neck that forms during pinchoff (illustrated in Figure 4.12b). We again use equation (4.14), and again assume it is active for $\min\left(\frac{1}{2}\tau, 600\mu s\right)$ for each bubble. The same sampling method for η is also used.

Merging: For bubble coalescence events, a similar mechanism is proposed in [44]. When two bubbles merge, very strong surface tension is generated at the merge point. This causes the bubble surface to expand, rapidly increasing the volume of the new bubble (see Figure 4.12c). Their model is

$$f(t,R) = \frac{6\sigma\kappa p_{in,0}}{\rho R^3}t^2, \tag{4.15}$$

where again we have multiplied their original equation by ρR^2 to transform to

pressure. The time that this forcing function is active is labeled t_{lim}, and is defined as the amount of time it takes for the expanding radius to reach a fixed fraction (sampled uniformly between $[0.4, 0.8]$) of the smaller bubble radius. An arbitrary modulation function is also added, resulting in the forcing function

$$f(t, R) = \left[\frac{1}{2} - \frac{1}{\pi}\tan^{-1}\left(3\frac{t - t_{lim}}{t_{lim}}\right)\right]\frac{6\kappa\sigma p_0}{\rho R^3}t^2.$$

We again limit the forcing to $\min(t_{lim}, 600\mu s)$.

4.8 Sound Synthesis

The last part of our pipeline (after fluid simulation, bubble identification and tracking, frequency estimation, and radiation analysis) is to synthesize the resulting sound at the listening position. While conceptually similar to [170], there are several different and important details in our approaches.

4.8.1 Audio Synthesis Details

Culling silent bubbles: To avoid unnecessary frequency and radiation solves for inaudible bubbles, we cull silent bubbles. Specifically, we do not perform any solves for bubbles that are older than $-ln(.01)/\beta$, where β is estimated from the bubble's equivalent spherical radius and Minnaert frequency.

Oscillator tracking: To avoid discontinuities in the synthesized sound, we continue oscillators through split and merge events. During split events, the parent bubble's oscillator continues to the largest child bubble. During merge events, the largest parent bubble's oscillator continues to the child bubble. We use the

standard RK4 method to integrate the oscillator equations, and did not need any special treatment to handle abrupt frequency changes.

Simultaneous events: A single bubble can undergo multiple events during a single timestep of our simulation. For example, a near-surface bubble could split and one of the daughter bubbles could touch the surface and disappear. Similarly, a bubble could be entrained and merge with another bubble during one time step. To handle these volume changes and missed events correctly, we monitor each bubble's volume during bubble tracking. If there is a sudden increase, we add a new bubble entrainment event and merge it immediately. When there is a sudden volume decrease, we add a split event followed immediately by a collapse event.

Frequency Extension

For various reasons such as resolution limits or the idealized damping model, simulated bubbles may reach the fluid surface and collapse before they are done oscillating. This singularity can limit our ability to resolve the bubble's chirp-like frequency response. To avoid such artifacts, we fit a small exponential model, ae^{ct}, to the frequency samples, and use it to extrapolate the bubble's frequency in time (see Figure 4.13). Parameters a and c are calculated to ensure C_0 and C_1 continuity.

Bubble Popping Sound Model

Bubbles that "pop" at the surface have their oscillators die out, but in reality there is a characteristic chirp-like popping sound due to the small pressurized

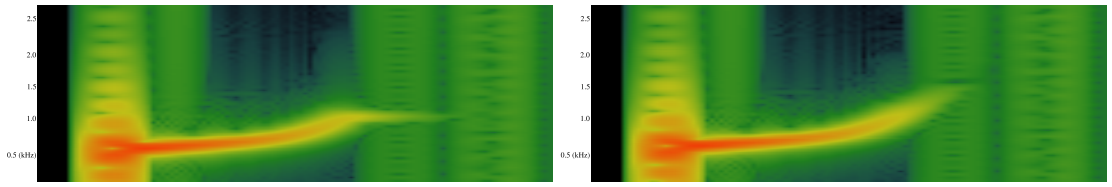


Figure 4.13: **Bubble frequency extension model:** (Left) a bubble collapses before it has finished oscillating, resulting in an audible sample-and-hold frequency artifact. (Right) To improve the approximation we extrapolate the frequency using a fitted exponential.

cavity that briefly rings like a so-called Helmholtz resonator [145]. These sounds are quiet, but occupy a part of the audio spectrum which is otherwise quiet in our model due to the predominantly lower pitch of bubbles resolvable by our fluid simulator. The physical process of bubble bursting and aerosol generation is actually terribly complicated [95]. We use a simplified model for the sound produced by bursting bubbles proposed for remote acoustic sensing of important bubble properties, such as bubble cap film thickness [48].

The model gives the time dependent frequency of a bubble pop as (equation 5 in [48])

$$f_H(t) = \frac{c}{2\pi}\sqrt{\frac{3\pi^2}{16V}R\sin\left(\frac{ut}{R}\right)}, \qquad 0 \le t \le t_{max},$$

where u is the velocity of the retracting bubble film, R is the bubble cap radius, and V is the bubble volume. t_{max} is the time it takes for the film to retract fully. The film retraction velocity can be estimated from the length of time the film has been draining for before it nucleates. We define the minimum drain time t_{min} as the time it would take for the film thickness to reach $R/10$.

When a bubble reaches the surface in our simulation, we uniformly sample the drain time between $[t_{min}, 20ms]$ to define u and synthesize a cosine chirp with frequency $f_H(t)$. We modulate the pop sound with an ad-hoc function chosen to

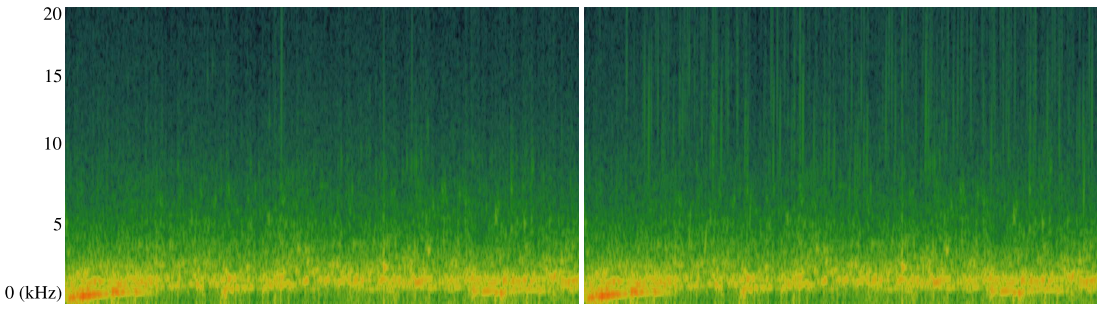


Figure 4.14: **Bubble popping sounds** add additional high-frequency content as shown here by comparing spectrograms of sounds produced (Left) without and (Right) with the popping sound model based on [Deane 2013]. The simulation example is the **pouring faucet**.

match experimental data from [48], namely,

$$mod(t) = \exp\left(\frac{\ln(.00001)}{t_{max}}t\right)\frac{2}{\pi}\operatorname{atan}\left(\frac{t}{2t_{max}}\right).$$

The amplitude of the pop is not well defined. Given the maximum absolute value of the corresponding bubble sound s_{max}, we choose to scale the pop sound so that it has a maximum magnitude of as_{max}, where a is uniformly sampled between $[.001, .03]$. The effect of the popping sound model is shown in Figure 4.14.

Bubble Bandwidth-Extension Scheme

Given the resolution limits of our fluid simulator, bubbles below a certain length scale can not be resolved correctly, resulting in a band-limited frequency response of the bubble oscillator model. Experimental studies of bubble populations in breaking waves have established various bubble size statistics, and have shown that the number of tiny bubbles tends to follow power-law models [50]. To artificially extend the frequency response of our renderings, we optionally seed audio-domain bubble events from a power-law distribution as follows.

We sample tiny bubbles in the audio domain, based on simulated larger bubbles. Specifically, for each entrained simulation bubble with radius $r_{parent} \geq$

2mm, we assume that the impact which created this bubble also generated other smaller bubbles with radii $r_{tiny} \in [0.1mm, 1mm]$. The number of artificial bubbles generated for each simulation bubble is uniformly sampled between $[0, 3000 * r_{parent}]$. Given the simulation bubble's creation time t, the start times for each of the artificial bubbles are uniformly sampled from $[t - 0.1, t]$ (because the tiny bubbles are created during the impact). The sizes of the tiny bubbles are sampled from a -3/2 power law, consistent with observations of the distribution of bubbles below the Hinze scale. Finally, since we have no geometry or positions for these artificial bubbles, we base their amplitude on the parent bubble. Given the parent bubble's transfer magnitude p_{parent}, we set the transfer value for a tiny bubble to 50 p_{parent} $(r_{tiny})^{1/3}$

4.8.2 Sound Synthesis Summary

In summary, our sound synthesis pipeline is very similar to previous work, with the important details that we need to track oscillators through split and merge events to avoid discontinuities, and simultaneous events need to be treated correctly. We proposed several audio domain methods to help add missing detail from our simulations, including frequency extension, a bubble popping model, and a microbubble model. The three latter models are all optional.

4.9 Results and Discussion

Please see our accompanying video for visual and audio results.

Results were computed on a heterogeneous cluster of 31 nodes, where

Example	Domain size (cm)	Length (s)	Simulation time (hours / cores)	# of bubbles / # of solves	Frequency solve time (hours)	Amortized speedup	Radiation solve time (hours)	Proxy evaluation time (hours)	% culled
Dripping Faucet	8 x 8 x 24	9.0	97 / 32	153 / 965	.005	0.81	.07	0.09	6.4
Pouring Faucet	8 x 8 x 24	8.5	402 / 64	331521 / 585311	23	7.35	52	3.8	71.2
Water Step	8 x 24 x 24	4.5	1000 / 96	420134 / 483654	20	4.85	44	20.0	71.8
Dam Break	16 x 16 x 32	2.64	394 / 64	114471 / 121646	2.9	4.72	15	0.66	72.4
Armadillo Drop	16 x 16 x 32	4.0	293 / 64	13981 / 11653	.245	3.82	3.4	0.58	87.9

Table 4.1: **Results.** Our fluid simulations used different numbers of cores, which are reported above. The frequency and radiation solves are massively parallel, and were computed using 680 cores. Proxy transfer evaluation was done on a single core, but could be parallelized easily.

each node's core count ranged from 8 to 64 (Intel Xeon X5355 and Xeon X7560 processors), as well as on the NSF cluster SuperMIC (two Intel Xeon E5-2680 processors per node). We report runtimes and simulation statistics in Table 4.1. The modifications we made to the *Gerris* code (to enable bubble tracking) are available on the project website `http://www.cs.cornell.edu/projects/Sound/bubbles`, along with our simulation scripts, code to interface with BEM++, and sound synthesis code.

4.9.1 Discussion of Tank Effects

Container effects such as reverberance can be significant. An example is shown in Figure 4.15. When a single bubble is entrained in a glass fish tank, strong echo can be seen in the air microphone signal. However, the sound recorded by a hydrophone of the same event looks much closer to a theoretical damped harmonic oscillator.

4.9.2 Validation

We performed several experiments to validate our frequency model.

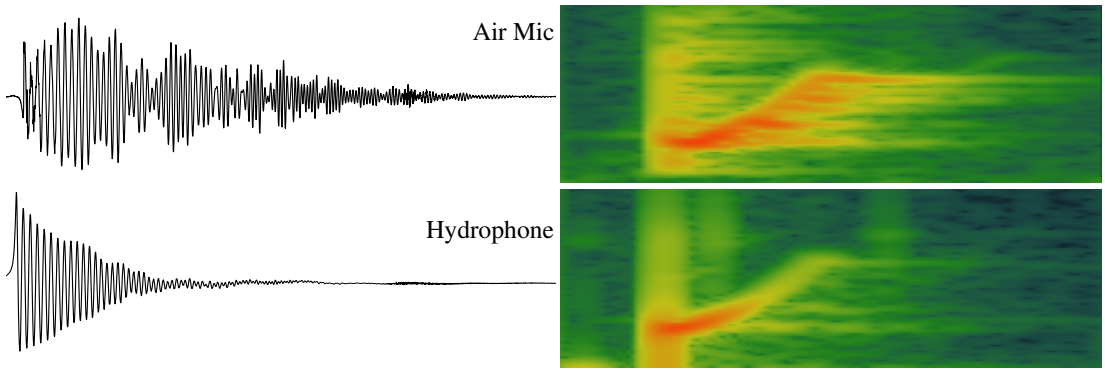


Figure 4.15: **Container effects:** Container effects can be strong in the air (top), while in a simultaneous hydrophone recording (bottom) the waveform of the same entrained bubble is much cleaner. Resonances of the container can be seen as lines in the spectrum, which continue after the bubble has popped.

Single bubble entrainment: We recorded several bubbles entrained by droplets, and simulated a similar entrainment case. We are able to capture the characteristic frequency chirp of the bubble as it rises (see Figure 4.16).

Underwater bubble creation: Using a syringe and plastic tubing, we also released underwater bubbles and recorded their emissions. As the bubble moves away from the tube (rigid surface), there is a slight pitch increase, but we do not see the characteristic chirp because the bubble finishes vibrating before it reaches the surface. A simulated scenario of a bubble moving away from an underwater tube produces a similar effect.

4.9.3 Large Results

Dripping faucet: Our dripping faucet example has a relatively small number of bubbles. It clearly demonstrates the pitch rise as bubbles approach the surface, as well as the benefit of our frequency extension model.

Dam break: There is a loud, low frequency sound produced by the large tubular

bubble in the dam break example. The sloshing sounds in this example are convincing. The dam break clearly demonstrates the benefit of our microbubble model, which adds higher frequency texture.

Water step: The water step has the most bubbles of all our simulations. While it has a fairly constant sound spectrum, the differences of our multiple models can be seen. This example also demonstrates the importance of transfer, as it sounds very dissonant when rendered without transfer.

Pouring faucet: We use the same domain as the dripping faucet, but this time use a constant 1cm radius stream to fill the container. This example most clearly shows the importance of transfer, allowing us to capture the characteristic pitch shift as the container fills.

Armadillo drop: For our last example, we dropped a water shaped armadillo into a pool of water. While this example is fun, it also highlights some of the deficiencies of our system. Even with our microbubble model, there is not much of an impact sound during the initial impact of the armadillo. A microbubble model based on droplet impacts, instead of on larger bubbles entrainment times, may help.

4.10 Conclusion

We have explored many stages of audiovisual fluid simulation, and identified the need for, and proposed, numerous sound simulation models. We believe that increased resolution of liquid sound generation mechanisms will also lead to improved visual fidelity of fluids in computer animation.

One surprising finding of our bubble frequency model was the large variations predicted based on spatial proximity to boundaries (see Figure 4.4), whereas the nonspherical pitch variations were comparatively more modest and less perceptually important for sound rendering. As Figure 4.7 demonstrates, there were some stong pitch variations due to shape, but these seemed rare. Perhaps the most important aspect of nonspherical bubbles was their ability to conform to the fluid-air interface and produce large pitch increases for rising bubbles due to the thin-plate capacitor effect (see Figure 4.6).

Our frequency model provides a way to capture the complex frequency effects that acoustic bubbles exhibit, and we hope that it will lead to better audiovisual simulation of water in the future, as well as more accurate methods for passive acoustic sensing. Our frequency model can be solved efficiently, and is surprisingly robust to coarse meshes (see Figure 4.8 top).

Limitations and future work: Far from solving the problem outright, the current study identifies many challenges, limitations, and opportunities for future work in realistic audiovisual fluid simulation. Our formulations calculate frequency and radiation based on an independent bubble assumption; however, bubbles can affect each other to produce frequency-coupled vibrations and in turn change their acoustic emissions. Our frequency-domain transfer model can capture resonance effects in containers and thereby improve the sound quality over previous work [170, 110]. But it misses perceptually significant time-domain reverberation effects. In addition, the container sound is missing, which can be an important source of fluid-solid coupling sounds. Also, we only model the dependence of ω on local geometry, but bubble damping β may also be affected. Acoustic radiation, while perceptually crucial, is computationally ex-

pensive. Our bubble-plane proxy is one attempt at performance improvement, but other techniques such as fast-multipole methods, FDTD with PML on the GPU, or geometric acoustics are interesting future work directions.

Our sound model is inherently band-limited due to resolution-limited fluid simulation. Post-processing effects such as popping and microbubble models can help. But it is also desired to incorporate more advanced multi-scale models in our system. The bubble forcing models we use are relatively simple, typically only capturing asymptotic forcing information, but assume idealized geometry and involve undetermined parameters. It is therefore interesting to explore data-driven approaches for automatically extracting forcing models and their parameters. Laboratory experiments reveal a wide-range of surface related sound events which do not radiate effectively underwater (as measured by a hydrophone), but that produce significant high-frequency sound contributions. Methods for approximating these processes, either in the geometric domain or audio domain, are necessary to improve sound realism. Finally, there are numerous important potential applications of fluid-sound synthesis in other fields of science and engineering that should be explored.

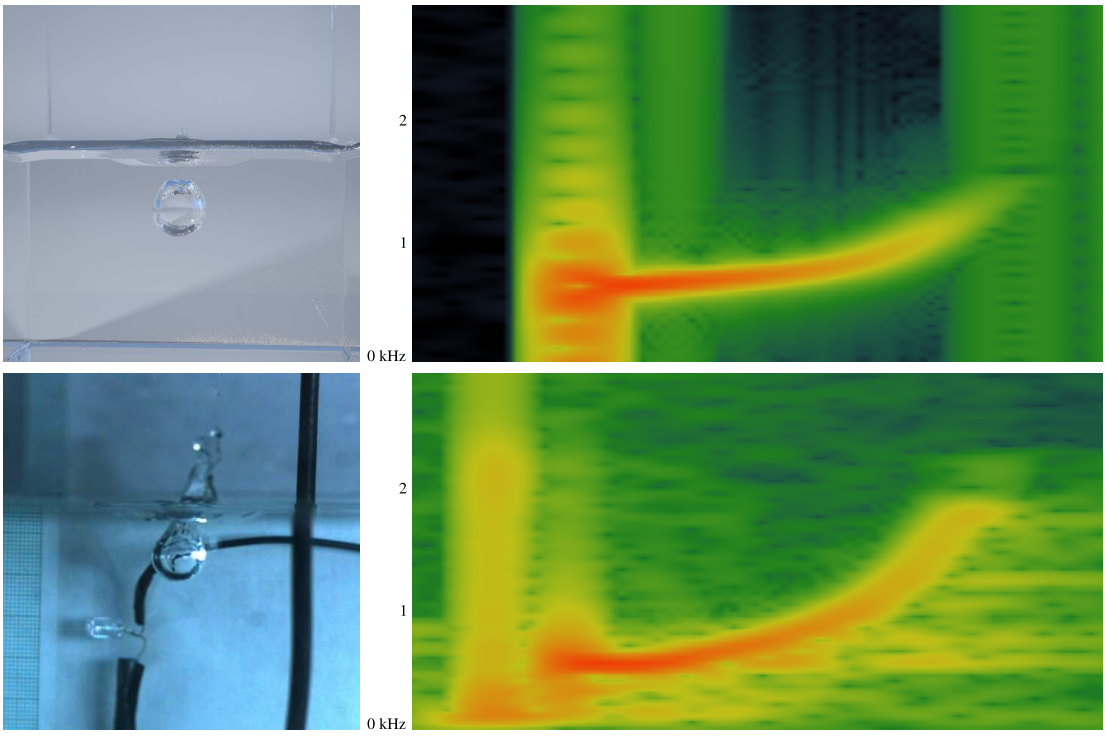


Figure 4.16: **Single Bubble Entrainment:** A simulated bubble entrainment event (top) produces a similar spectrum to a recorded entrainment event (bottom).

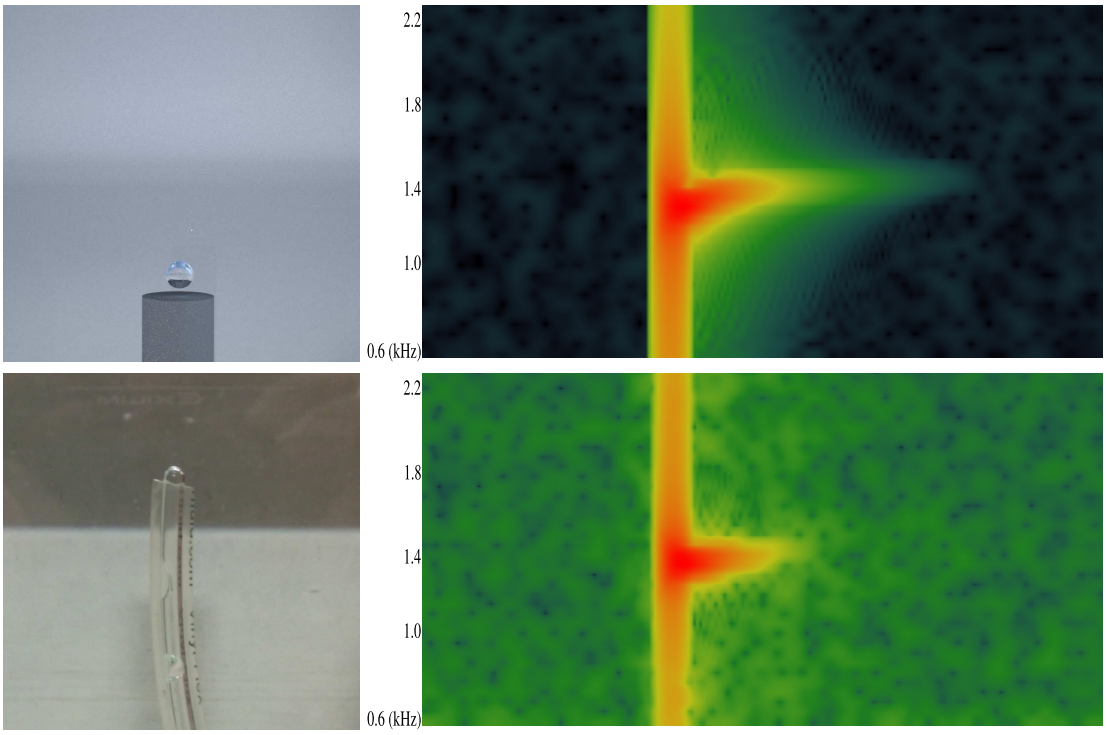


Figure 4.17: **Underwater bubble release:** A bubble released from an underwater tube shows a slight frequency rise as it moves away from the rigid tube. The simulation (top) matches well with experiment (bottom).

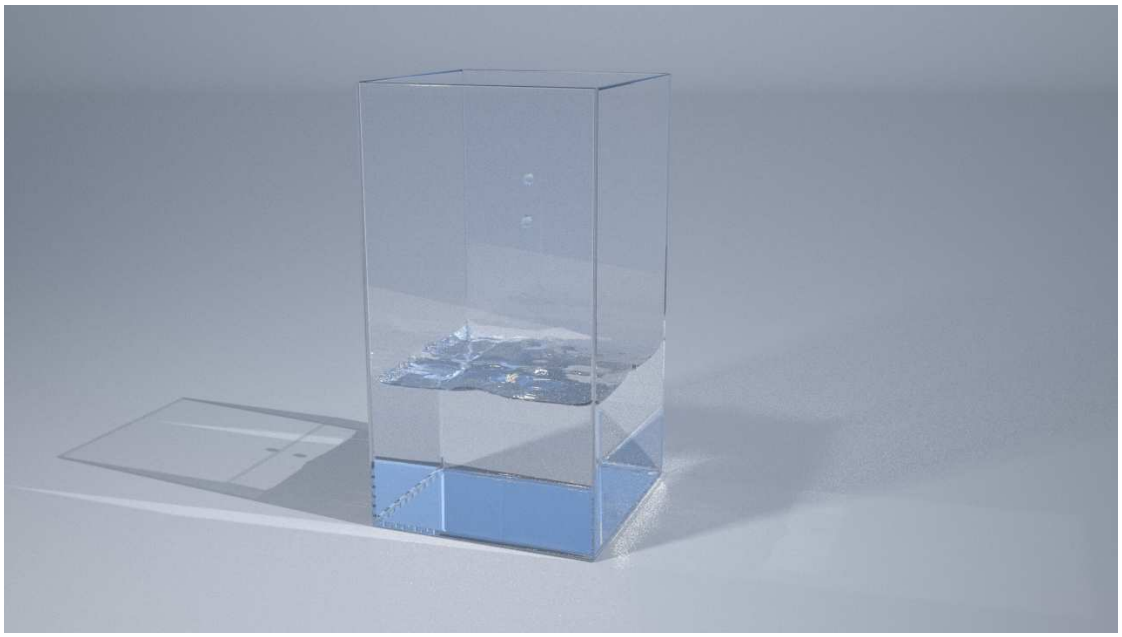


Figure 4.18: **Dripping Faucet**

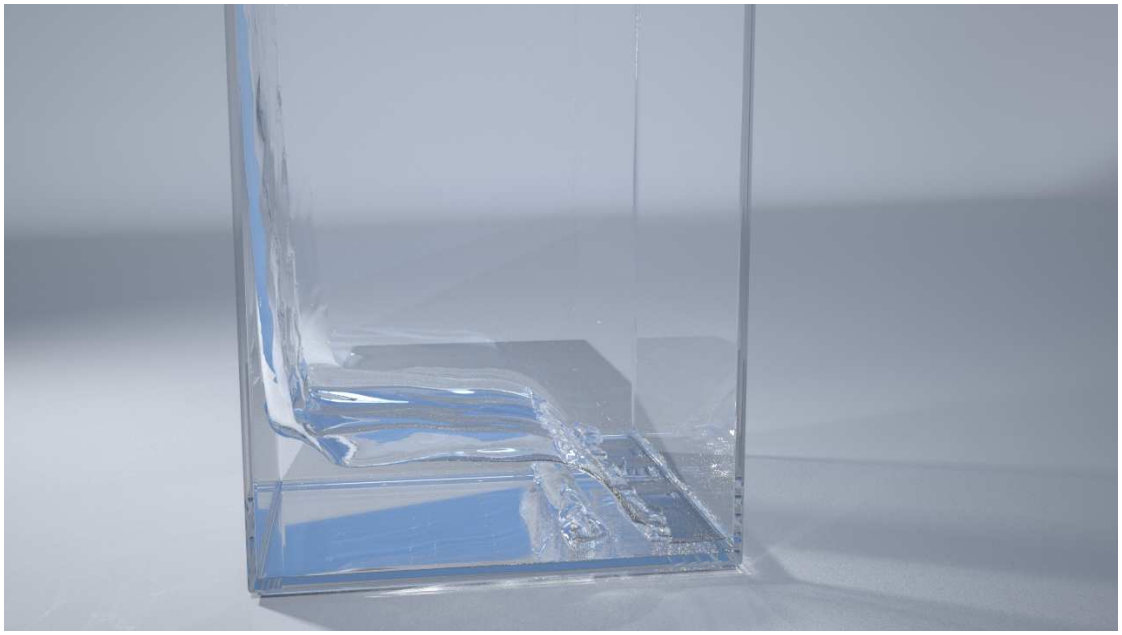


Figure 4.19: **Dam Break**

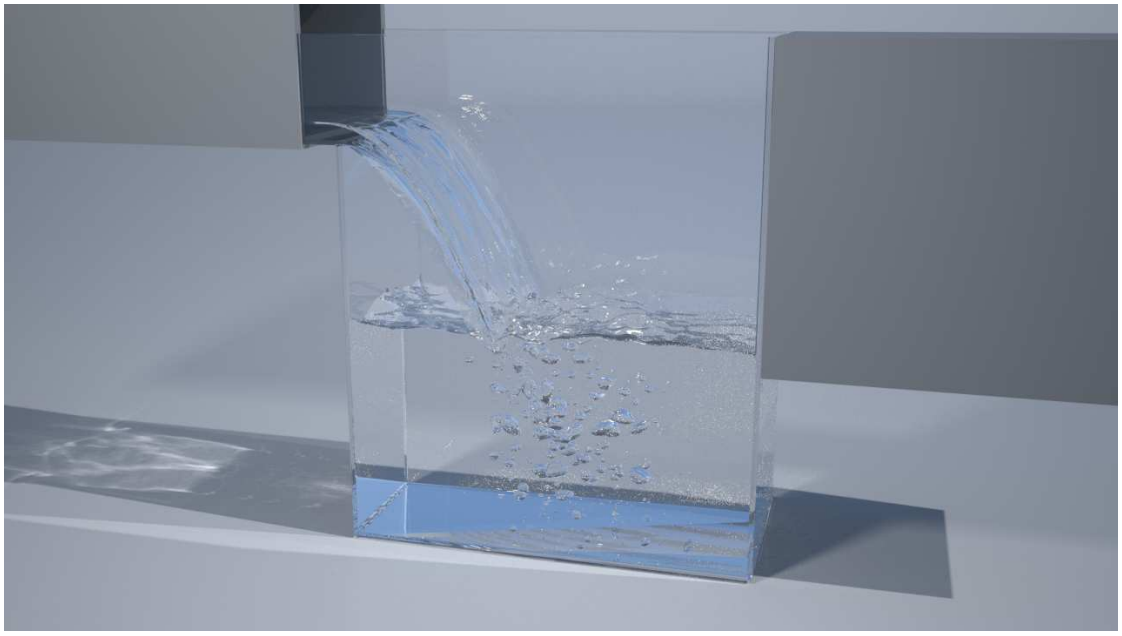


Figure 4.20: **Water Step**

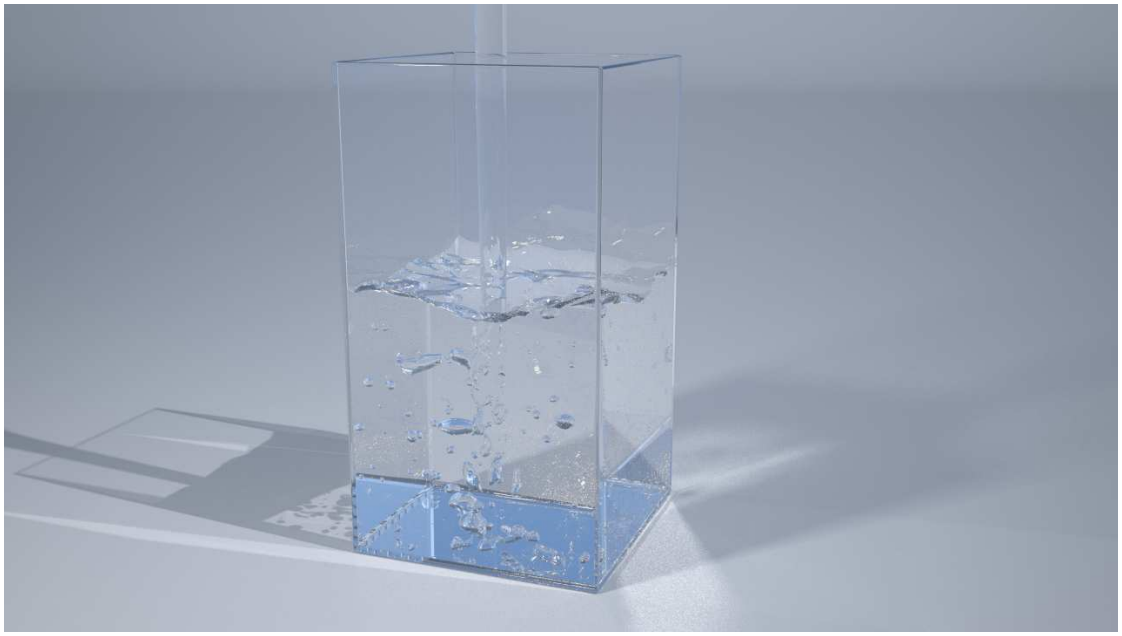


Figure 4.21: **Pouring Faucet**

Figure 4.22: **Armadillo**

CHAPTER 5

CONCLUSION

5.1 Summary and Conclusions

Physical simulation has become an important tool in computer graphics as a method to add detail to animations while reducing artist labor. As better methods are developed and hardware capabilities improve, our ability to use simulation as a general problem solving tool increases.

This thesis investigates several ways of using physical simulation for problem solving, and also steps to make physical simulation easier to use. In Chapter 2, we explored the inverse problem of synthesizing an animation aligned with a recorded sound. This allows us to leverage the realism of recorded sounds, and provides a form of motion control. Our main insight was to sample a large number of simulations. In Chapter 3, we made a specific type of simulation (modal sound synthesis) easier to use by reducing its memory requirements. We fit moving least squares approximation to mode shapes, in a perceptually meaningful way, and were able to achieve compression ratios of up to 1000x. Finally, in Chapter 4, we developed a better model of bubble frequency, and used incompressible, two-phase fluid simulation to simulate water sounds. This helps to create more realistic sound, while also illuminating how well current bubble forcing and damping models work.

5.2 Limitations and Future Work

In addition to the limitations and future work mentioned in previous chapters, there are several high level problems and goals for physical simulation in computer graphics. It is often necessary to sacrifice accuracy for speed. Two aspects of this could be done better. First, more verification against real world experiments, and clarification of limitations, is necessary. We performed several verification experiments of our bubble frequency model in Chapter 4, however these are fairly rudimentary. Second, human perception should be dealt with more rigorously. Often the metric for success of a computer graphics simulation method is "does it look good?". While this can result in interesting results, it creates challenges when attempting to evaluate how well a certain method will work in a new scenario, and can result in subjective evaluation. In Chapter 3 we demonstrated one way to leverage published results in order to incorporate human perception more rigorously.

APPENDIX A

APPENDIX FOR CHAPTER 3

A.1 Derivation of relative mode importance

We now derive the expression for the relative average far-field pressure as proportional to $\|\mathbf{u}^j\|\,\|\mathbf{c}^j\|_F/(\omega^j)^2$ used when allocating relative mode errors. Given the pressure is $O(|p||q|)$, we estimate $|q|$ using the modal response to an impulse $\mathbf{f}\,\delta(t)$,

$$q(t) = \tfrac{1}{\omega} e^{-\frac{\alpha+\beta\omega^2}{2}t} \sin(\omega t)(\mathbf{u} \bullet \mathbf{f}) \implies |q| \le \tfrac{|\mathbf{u}\bullet\mathbf{f}|}{\omega} \overset{\|\mathbf{f}\|=1}{\le} \tfrac{\|\mathbf{u}\|}{\omega}.$$

We estimate $O(|p|)$ as proportional to the square-root of (far-field) radiated power. Consider a unit-amplitude vibration of a single mode, with acoustic transfer $p(\boldsymbol{x})$ of (3.2). The time-averaged radiated power through a surrounding sphere S_R of radius R, is proportional to $P_R = \int_{S_R} |p|^2\, dS = \int_{S_R} |p(\boldsymbol{x})|^2 \sin\theta\, R^2\, d\theta d\phi$ [74]. For the far-field power ($R \to \infty$) only leading-order $1/r$ contributions from p contribute. Since the Hankel functions have $h_n^{(2)}(kr) \sim i^{n+1} e^{-ikr}/kr \quad \text{as } kr \to \infty$, and using the orthogonality of the spherical harmonics, we find that

$$P_\infty = \tfrac{1}{k^2} \textstyle\sum_{n=0}^{\bar{n}} \sum_{m=-n}^{n} |c_n^m|^2 = \tfrac{1}{k^2} \|\mathbf{c}\|_F^2.$$

We then estimate the mode-relative amplitude of the space- and time-averaged far-field transfer as $|p| = \sqrt{P_\infty} \sim \|\mathbf{c}\|_F/k$. It follows that the relative modal importance is given by $\|\mathbf{u}^j\|\,\|\mathbf{c}^j\|_F/(\omega^j)^2$.

APPENDIX B

APPENDIX FOR CHAPTER 4

B.1 Fast Amortized BEM Solver Details

Here we provide details (from §4.5.4) on how to efficiently evaluate $\boldsymbol{x} = \mathbf{A}^{-1}\mathbf{b}$ in our frequency solver for the common case of multiple bubbles (with $n_b \ll n_a + n_b$). First, to clarify, we avoided the expensive inverse in the U block of $\mathbf{A}^{-1}$ expression (4.12) using the Sherman-Morrison-Woodbury low-rank update formula [66] to obtain

$$U = \left(D - CG_{bb}^{-1}B\right)^{-1} \tag{B.1}$$

$$= D^{-1} + D^{-1}C\underbrace{(G_{bb} - BD^{-1}C)^{-1}}_{X}BD^{-1} \tag{B.2}$$

$$= D^{-1}(I + C\underbrace{XBD^{-1}}_{-Y}) = D^{-1}(I - CY). \tag{B.3}$$

The bubble-independent solver setup constructs the air-solid domain form-factor matrix D, and the LU factorization for D^{-1}. Then given a specific bubble geometry, we construct the remaining blocks of A, and then compute

$$G_{bb}^{-1} \leftarrow LUSolver(G_{bb}) \tag{B.4}$$

$$T_1 \leftarrow D^{-1}C \qquad (n_b \text{ LU solves for } C \text{ RHS}) \tag{B.5}$$

$$X \leftarrow LUSolver(G_{bb} - BT_1). \qquad (\text{then discard } T_1) \tag{B.6}$$

Then given the RHS vector (in block form) $\mathbf{b} = \begin{pmatrix} b_b \\ b_{as} \end{pmatrix}$ we can form $\boldsymbol{x} = \mathbf{A}^{-1}\mathbf{b} = \begin{pmatrix} x_b \\ x_{as} \end{pmatrix}$ as follows. Since

$$x_b = Xb_b + Yb_{as} = X(b_b - BD^{-1}b_{as}), \tag{B.7}$$

we can form x_b via (here t_* are temporary variables)

$$t_1 \leftarrow D^{-1} b_{as} \qquad \text{(LU solve)} \tag{B.8}$$

$$t_2 \leftarrow b_b - B t_1 \tag{B.9}$$

$$\boxed{x_b} \leftarrow X t_2. \qquad \text{(LU solve)} \tag{B.10}$$

Second, observe that

$$x_{as} = Z b_b + U b_{as} = U(-C G_{bb}^{-1} b_b + b_{as}) \tag{B.11}$$

so we can form $x_{as} = U t_2 = D^{-1}(I - CY) t_2$ as

$$t_1 \leftarrow G_{bb}^{-1} b_b \qquad \text{(LU solve)} \tag{B.12}$$

$$t_2 \leftarrow b_{as} - C t_1 \tag{B.13}$$

$$t_3 \leftarrow Y t_2 = -X(B(D^{-1} t_2))) \qquad \text{(2 LU solves)} \tag{B.14}$$

$$t_4 \leftarrow t_2 - C t_3 \tag{B.15}$$

$$\boxed{x_{as}} \leftarrow D^{-1} t_4 \qquad \text{(LU solve)} \tag{B.16}$$

Assuming (for simplicity) that $n_b = O(1)$ and $N = n_a + n_s$, then it involves only $O(N)$ new form factor computations per bubble (instead of $O(N^2)$), and it costs $O(N^2)$ to construct this A^{-1} transformation (instead of $O(N^3)$), and $O(N^2)$ to apply it to form $A^{-1} b$. The main per-bubble cost is $n_b + 3$ applications of the D^{-1} LU solver.

BIBLIOGRAPHY

[1] Jean-Marie Adrien. Representations of musical signals. chapter The Missing Link: Modal Synthesis, pages 269–298. MIT Press, Cambridge, MA, USA, 1991.

[2] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `https://code.google.com/p/ceres-solver/`.

[3] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point set surfaces. In *Proceedings of the Conference on Visualization '01*, VIS '01, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society.

[4] Marc Alexa and Wolfgang Müller. Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418, 2000.

[5] Steven S. An, Doug L. James, and Steve Marschner. Motion-driven concatenative synthesis of cloth sounds. *ACM Transactions on Graphics*, 31(4):102:1–102:10, July 2012.

[6] Ryoichi Ando, Nils Thuerey, and Chris Wojtan. A stream function solver for liquid simulations. *ACM Transactions on Graphics (TOG)*, 34(4):53, 2015.

[7] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, July 2002.

[8] Kaoru Ashihara. Hearing thresholds for pure tones above 16khz. *The Journal of the Acoustical Society of America*, 122(3), 2007.

[9] Prasanta Kumar Banerjee and Roy Butterfield. *Boundary element methods in engineering science*, volume 17. McGraw-Hill London, 1981.

[10] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Comput. Graph. (Proc. SIGGRAPH 89)*, 23(3):223–232, 1989.

[11] D. Baraff. Coping with friction for non-penetrating rigid body simulation. *Comput. Graph. (Proc. SIGGRAPH 91)*, 25(4):31–40, 1991.

[12] D. Baraff and A. Witkin. Large steps in cloth simulation. In *ACM SIGGRAPH 98*, pages 43–54. ACM Press/ACM SIGGRAPH, 1998.

[13] Jernej Barbič, Marco da Silva, and Jovan Popović. Deformable object animation using reduced optimal control. *ACM Trans. on Graphics (SIGGRAPH 2009)*, 28(3), 2009.

[14] Jernej Barbič and Doug James. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Transactions on Graphics*, 24(3):982–990, August 2005.

[15] Jernej Barbič, Funshing Sin, and Eitan Grinspun. Interactive editing of deformable simulations. *ACM Trans. on Graphics (SIGGRAPH 2012)*, 31(4), 2012.

[16] Ronen Barzel, John F. Hughes, and Daniel N. Wood. Plausible motion simulation for computer graphics animation. In *EGCAS '96: Seventh International Workshop on Computer Animation and Simulation*, pages 183–197, 1996.

[17] Klaus-Jürgen Bathe. *Finite Element Procedures*. Prentice Hall, second edition, 1996.

[18] C. Belta and V. Kumar. An SVD-based projection method for interpolation on SE(3). *Robotics and Automation, IEEE Transactions on*, 18(3):334–345, 2002.

[19] Jan Bender, Kenny Erleben, and Jeff Trinkle. Interactive simulation of rigid body dynamics in computer graphics. In *Computer Graphics Forum*. Wiley Online Library, 2013.

[20] Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. Discrete viscous threads. *ACM Transactions on Graphics (TOG)*, 29(4):116, 2010.

[21] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. Discrete elastic rods. *ACM Trans. Graph.*, 27(3):63:1–63:12, August 2008.

[22] Gaurav Bharaj, David I. W. Levin, James Tompkin, Yun Fei, Hanspeter Pfister, Wojciech Matusik, and Changxi Zheng. Computational design of metallophone contact sounds. *ACM Trans. Graph.*, 34(6):223:1–223:13, October 2015.

[23] Kiran S Bhat, Steven M Seitz, Jovan Popović, and Pradeep K Khosla. Com-

puting the physical parameters of rigid-body motion from video. In *Computer Vision—ECCV 2002*, pages 551–565. Springer, 2002.

[24] G. R. Bigg, T. D. Jickells, P. S. Liss, and T. J. Osborn. The role of the oceans in climate. *International Journal of Climatology*, 23(10):1127–1159, 2003.

[25] Nicolas Bonneel, George Drettakis, Nicolas Tsingos, Isabelle Viaud-Delmon, and Doug James. Fast Modal Sounds with Scalable Frequency-Domain Synthesis. *ACM Transactions on Graphics*, 27(3):24:1–24:9, August 2008.

[26] Landon Boyd and Robert Bridson. MultiFLIP for energetic two-phase fluid simulation. *ACM Transactions on Graphics (TOG)*, 31(2):16, 2012.

[27] Sir William Henry Bragg. *The World of Sound*. G. Bell and Sons Ltd., 1920.

[28] Robert Bridson. *Fluid simulation for computer graphics*. CRC Press, 2008.

[29] Armin Bruderlin and Lance Williams. Motion signal processing. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 97–104, August 1995.

[30] Oleksiy Busaryev, Tamal K Dey, Huamin Wang, and Zhong Ren. Animating bubble interactions in a liquid foam. *ACM Transactions on Graphics (TOG)*, 31(4):63, 2012.

[31] M. Cardle, L. Barthe, S. Brooks, and P. Robinson. Music-driven motion editing: local motion transformations guided by music analysis. *Proceedings 20th Eurographics UK Conference*, pages 38–44, 2002.

[32] Jeffrey N. Chadwick, Steven S. An, and Doug L. James. Harmonic Shells: A Practical Nonlinear Sound Model for Near-Rigid Thin Shells. *ACM Transactions on Graphics*, 28(5):119:1–119:10, December 2009.

[33] Jeffrey N. Chadwick and Doug L. James. Animating fire with sound. *ACM Transactions on Graphics*, 30(4):84:1–84:8, July 2011.

[34] Jeffrey N. Chadwick, Changxi Zheng, and Doug L. James. Faster acceleration noise for multibody animations using precomputed soundbanks. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 265–273, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[35] Jeffrey N. Chadwick, Changxi Zheng, and Doug L. James. Precomputed acceleration noise for improved rigid-body sound. *ACM Transactions on Graphics*, 31(4):103:1–103:9, July 2012.

[36] Stephen Chenney and D. A. Forsyth. Sampling plausible solutions to multi-body constraint problems. *Proceedings of the 27th annual conference on*, pages 219–228, 2000.

[37] R. Chicharro and a. Vazquez. The acoustic signature of gas bubbles generated in a liquid cross-flow. *Experimental Thermal and Fluid Science*, 55:221–227, 2014.

[38] Michel Chion. *Audio-Vision: Sound on Screen*. Columbia University Press, 1994.

[39] Alexander Clegg, Jie Tan, Greg Turk, and C. Karen Liu. Animating human dressing. *ACM Trans. Graph.*, 34(4):116:1–116:9, July 2015.

[40] Michael F. Cohen. Interactive spacetime control for animation. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 293–302, New York, NY, USA, 1992. ACM.

[41] Perry Cook. *Real Sound Synthesis for Interactive Applications*. A.K. Peters, 2002.

[42] Perry R Cook. Integration of physical modeling for synthesis and animation. In *Proceedings of the International Computer Music Conference*, pages 525–528, 1995.

[43] Stelian Coros, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert Sumner, and Markus Gross. Deformable objects alive! *ACM Trans. Graph.*, 31(4):69:1–69:9, July 2012.

[44] Helen Czerski. A candidate mechanism for exciting sound during bubble coalescence. *The Journal of the Acoustical Society of America*, 129(3):EL83–EL88, 2011.

[45] Helen Czerski and Grant B. Deane. Contributions to the acoustic excitation of bubbles released from a nozzle. *The Journal of the Acoustical Society of America*, 128(5):2625–2634, 2010.

[46] Helen Czerski and Grant B. Deane. The effect of coupling on bubble

fragmentation acoustics. *The Journal of the Acoustical Society of America*, 129(1):74–84, 2011.

[47] Fang Da, Christopher Batty, Chris Wojtan, and Eitan Grinspun. Double bubbles sans toil and trouble: discrete circulation-preserving vortex sheets for soap films and foams. *ACM Transactions on Graphics (TOG)*, 34(4):149, 2015.

[48] Grant B Deane. Determining the bubble cap film thickness of bursting bubbles from their acoustic emissions. *The Journal of the Acoustical Society of America*, 133(February 2013):EL69–75, 2013.

[49] Grant B. Deane and Helen Czerski. A mechanism stimulating sound production from air bubbles released from a nozzle. *The Journal of the Acoustical Society of America*, 123(6):EL126–EL132, 2008.

[50] Grant B Deane and M Dale Stokes. Scale dependence of bubble creation mechanisms in breaking waves. *Nature*, 418(6900):839–844, 2002.

[51] Grant B. Deane and M. Dale Stokes. Model calculations of the underwater noise of breaking waves and comparison with experiment. *The Journal of the Acoustical Society of America*, 127(6):3394–3410, 2010.

[52] Li Ding and David M Farmer. Observations of breaking surface wave statistics. *Journal of Physical Oceanography*, 24(6):1368–1387, 1994.

[53] Damien Jade Duff, Thomas Morwald, Rustam Stolkin, and Jeremy Wyatt. Physical simulation for monocular 3d model based tracking. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5218–5225. IEEE, 2011.

[54] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (TOG)*, 21(3):736–744, 2002.

[55] Kenny Erleben. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.*, 26(2), June 2007.

[56] Hugo Fastl and Eberhard Zwicker. *Psychoacoustics: Facts and models*, volume 22. Springer, 2006.

[57] B. Feldman, J. O'Brien, and B. Klingner. Animating gases with hybrid meshes. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24(3):904–909, 2005.

[58] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 24(3):544–552, July 2005.

[59] N. Foster and R. Fedkiw. Practical animation of liquids. *Proc. ACM SIGGRAPH*, pages 23–30, 2001.

[60] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 1996.

[61] Thomas Funkhouser, Ingrid Carlbom, Gary Elko, Gopal Pingali, Mohan Sondhi, and Jim West. A beam tracing approach to acoustic modeling for interactive virtual environments. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 21–32, July 1998.

[62] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[63] David Grelaud, Nicolas Bonneel, Michael Wimmer, Manuel Asselot, and George Drettakis. Efficient and practical audio-visual rendering for games using crossmodal perception. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 177–182, New York, NY, USA, February 2009. ACM.

[64] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, July 2003.

[65] Igor Guskov and Andrei Khodakovsky. Wavelet compression of parametrically coherent mesh sequences. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 183–192, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[66] William W Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989.

[67] Klaus Hildebrandt, Christian Schulz, Christoph von Tycowicz, and Konrad Polthier. Interactive spacetime control of deformable objects. *ACM Trans. Graph.*, 31(4):71:1–71:8, July 2012.

[68] Michael Hofer and Helmut Pottmann. Energy-minimizing splines in manifolds. *ACM Transactions on Graphics*, 23(3):284–293, August 2004.

[69] Jeong-Mo Hong and Chang-Hun Kim. Animation of bubbles in liquid. In *Computer Graphics Forum*, volume 22, pages 253–262. Wiley Online Library, 2003.

[70] Jeong-Mo Hong and Chang-Hun Kim. Discontinuous fluids. *ACM Transactions on Graphics (TOG)*, 24(3):915–920, 2005.

[71] M. S. Howe. *Acoustics of Fluid-Structure Interatctions*. Cambridge Press, 1998.

[72] M. S. Howe. *Theory of Vortex Sound*. Cambridge Press, 2002.

[73] M. S. Howe and N. A. A. Hagen. On the impact noise of a drop falling on water. *Journal of Sound and Vibration*, 330(4):625–635, 2011.

[74] Michael S Howe. *Acoustics of fluid-structure interactions*. Cambridge University Press, 1998.

[75] Doug L. James, Jernej Barbič, and Dinesh K. Pai. Precomputed Acoustic Transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics*, 25(3):987–995, July 2006.

[76] Doug L. James and Dinesh K. Pai. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph.*, 22(1):47–82, January 2003.

[77] Doug L. James, Christopher D. Twigg, Andrew Cove, and Robert Y. Wang. Mesh Ensemble Motion Graphs: Data-driven mesh animation with constraints. *ACM Transactions on Graphics*, 26(4):17:1—-17:16, October 2007.

[78] Jingyi Jin, Michael Garland, and Edgar A. Ramos. Mls-based scalar fields over triangle meshes and their application in mesh processing. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09, pages 145–153, New York, NY, USA, 2009. ACM.

[79] Byungmoon Kim, Yingjie Liu, Ignacio Llamas, Xiangmin Jiao, and Jarek Rossignac. Simulation of bubbles in foam with the volume control method. In *ACM Transactions on Graphics (TOG)*, volume 26, page 98. ACM, 2007.

[80] Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F. O'Brien. Near-exhaustive precomputation of secondary cloth effects. *ACM Transactions on Graphics*, 32(4):87:1–7, July 2013. Proceedings of ACM SIGGRAPH 2013, Anaheim.

[81] Tae-hoon Kim, Sang Il Park, and Sung Yong Shin. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics*, 22(3):392–401, 2003.

[82] Theodore Kim, David Adalsteinsson, and Ming C. Lin. Modeling ice dynamics as a thin-film stefan problem. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '06, pages 167–176, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[83] Theodore Kim and Ming C. Lin. Physically based animation and rendering of lightning. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, PG '04, pages 267–275, Washington, DC, USA, 2004. IEEE Computer Society.

[84] Zygmunt Klusek and Aliaksandr Lisimenka. Acoustic noise generation under plunging breaking waves. *Oceanologia*, 55(4):809–836, 2013.

[85] Krasimir Kolarov and William Lynch. Wavelet compression for 3d and higher-dimensional objects. In *Proc. of SPIE Conference on Applications of Digital Image Processing XX , Volume 3164*, 1997.

[86] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion Graphs. *ACM Transactions on Graphics*, 21(3):473–482, 2002.

[87] J.J. Kuffner. Effective sampling and distance metrics for 3D rigid body path planning. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3993–3998. IEEE, 2004.

[88] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, 26(3):57.1–57.10, July 2007. Special issue on Proceedings of SIGGRAPH 2007.

[89] Horace Lamb. *Hydrodynamics*. Cambridge University Press, 1932.

[90] Hyun-Chul Lee and In-Kwon Lee. Automatic synchronization of background music and motion in computer animation. *Computer Graphics Forum*, 24(3):353–361, 2005.

[91] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, July 2002.

[92] T. G. Leighton. *The Acoustic Bubble*. Academic Press, 1994.

[93] Tim Leighton, Dan Finfer, Ed Grover, and Paul White. An acoustical hypothesis for the spiral bubble nets of humpback whales and the implications for whale feeding. *Acoustic Bulletin*, 22(1):17–21, 2007.

[94] Thomas Lewiner, Hélio Lopes, Antônio Wilson Vieira, and Geovan Tavares. Efficient implementation of marching cubes cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, december 2003.

[95] H. Lhuissier and E. Villermaux. Bursting bubble aerosols. *Journal of Fluid Mechanics*, 696:5–44, 4 2012.

[96] Yaron Lipman, Xiaobai Chen, Ingrid Daubechies, and Thomas Funkhouser. Symmetry factored embedding and distance. *ACM Trans. Graph.*, 29(4):103:1–103:12, July 2010.

[97] Y. J. Liu. *Fast Multipole Boundary Element Method: Theory and Applications in Engineering*. Cambridge University Press, Cambridge, 2009.

[98] M. S. Longuet-Higgins. An analytical model of sound production by raindrops. *Journal of Fluid Mechanics*, 214:395–410, 1990.

[99] Michael S. Longuet-Higgins. Monopole emission of sound by asymmetric bubble oscillations. Part 1. Normal modes. *Journal of Fluid Mechanics*, 201:525–541, 1989.

[100] Michael S. Longuet-Higgins. Monopole emission of sound by asymmetric bubble oscillations. Part 2. An initial-value problem. *Journal of Fluid Mechanics*, 201:543–565, 1989.

[101] Michael S. Longuet-Higgins. Resonance in nonlinear bubble oscillations. *Journal of Fluid Mechanics*, 224:531, 1991.

[102] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. Multiple interacting liquids. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 812–819. ACM, 2006.

[103] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch, and François X. Sillion. Accurate detection of symmetries in 3d shapes. *ACM Trans. Graph.*, 25(2):439–464, April 2006.

[104] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM Trans. Graph.*, 23(3):449–456, August 2004.

[105] Steven L Means and Richard M Heitmeyer. Surf-generated noise signatures: a comparison of plunging and spilling breakers. *The Journal of the Acoustical Society of America*, 112(2):481–488, 2002.

[106] Herman Medwin and Matthew M. Beaky. Bubble sources of the Knudsen sea noise spectra. *The Journal of the Acoustical Society of America*, 86(3):1124–1130, 1989.

[107] M. Minnaert. On musical air-bubbles and the sounds of running water. *Philosophical Magazine*, 16(104):235–248, 1933.

[108] Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Trans. Graph.*, 25(3):560–568, July 2006.

[109] Niloy J. Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report*, 2012.

[110] William Moss, Hengchin Yeh, Jeong-Mo Hong, Ming C. Lin, and Dinesh Manocha. Sounding liquids: Automatic sound synthesis from fluid simulation. *ACM Trans. Graph.*, 29(3):21:1–21:13, July 2010.

[111] Michael B. Nielsen and Robert Bridson. Guide shapes for high resolution naturalistic liquid simulation. *ACM Trans. Graph.*, 30(4):83:1–83:8, July 2011.

[112] Jeffrey A Nystuen. Rainfall measurements using underwater ambient noise. *The Journal of the Acoustical Society of America*, 79(4):972–982, 1986.

[113] J. O'Brien and J. Hodgins. Graphical modeling and animation of brittle fracture. In *Proc. SIGGRAPH 99*, volume 18, pages 137–146, 1999.

[114] James F. O'Brien, Perry R. Cook, and Georg Essl. Synthesizing sounds from physically based motion. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 529–536, August 2001.

[115] James F. O'Brien, Chen Shen, and Christine M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 175–181, July 2002.

[116] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006.

[117] Carol O'Sullivan, John Dingliana, Thanh Giang, and Mary K. Kaiser. Evaluating the visual fidelity of physically based animations. *ACM Transactions on Graphics*, 22(3 (July)):527–536, July 2003.

[118] Dinesh K. Pai, Kees van den Doel, Doug L. James, Jochen Lang, John E. Lloyd, Joshua L. Richmond, and Som H. Yau. Scanning physical interaction behavior of 3d objects. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 87–96, New York, NY, USA, 2001. ACM.

[119] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 163–170, Washington, DC, USA, 2002. IEEE Computer Society.

[120] S. Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of Computational Physics*, 190(2):572–600, 2003.

[121] S. Popinet. An accurate adaptive solver for surface-tension-driven interfacial flows. *Journal of Computational Physics*, 228(16):5838—5866, 2009.

[122] J. Popović, S.M. Seitz, M. Erdmann, Z. Popović, and Andrew Witkin. Interactive manipulation of rigid body simulations. In *Proceedings of ACM*

SIGGRAPH 2000, pages 209–218. ACM Press/Addison-Wesley Publishing Co., 2000.

[123] Jovan Popović, Steven M. Seitz, and Michael Erdmann. Motion sketching for control of rigid-body simulations. *ACM Transactions on Graphics*, 22(4):1034–1054, October 2003.

[124] Andrea Prosperetti. Bubble dynamics in oceanic ambient noise. In BryanR. Kerman, editor, *Sea Surface Sound*, volume 238 of *NATO ASI Series*, pages 151–171. Springer Netherlands, 1988.

[125] Andrea Prosperetti. Bubble-related ambient noise in the ocean. *The Journal of the Acoustical Society of America*, 84(3):1042–1054, 1988.

[126] H.C. Pumphery, L.A. Crum, and L Bjørnø. Underwater sound produced by individual drop impacts and rainfall. *J. Acoust. Soc. Am.*, 85:1518–1526, 1989.

[127] H.C. Pumphrey and J.E. Ffowcs Williams. Bubbles as sources of ambient noise. *IEEE Journal of Oceanic Engineering*, 15(4):268–274, 1990.

[128] N. Raghuvanshi, N. Galoppo, and M.C. Lin. Accelerated wave-based acoustics simulation. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 91–102. ACM New York, NY, USA, 2008.

[129] Nikunj Raghuvanshi and Ming C. Lin. Interactive sound synthesis for large scale environments. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, I3D '06, pages 101–108, New York, NY, USA, 2006. ACM.

[130] Nikunj Raghuvanshi, Rahul Narain, and Ming C. Lin. Efficient and accurate sound propagation using adaptive rectangular decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):789–801, 2009.

[131] Lord Rayleigh. VIII. On the pressure developed in a liquid during the collapse of a spherical cavity. *Philosophical Magazine Series 6*, 34(200):94–98, 1917.

[132] Joshua L. Richmond. Automatic measurement and modeling of contact sounds. Master's thesis, University of British Columbia, April 2000.

[133] Alec R. Rivers and Doug L. James. Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph.*, 26(3), July 2007.

[134] Prashant Sachdeva, Shinjiro Sueda, Susanne Bradley, Mikhail Fain, and Dinesh K. Pai. Biomechanical simulation and control of hands and tendinous systems. *ACM Trans. Graph.*, 34(4):42:1–42:10, July 2015.

[135] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM Trans. Graph.*, 25(3):533–540, July 2006.

[136] Peter Schröder and Wim Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 161–172, New York, NY, USA, 1995. ACM.

[137] Jaewoo Seo, Geoffrey Irving, J. P. Lewis, and Junyong Noh. Compression and direct manipulation of complex blendshape models. *ACM Trans. Graph.*, 30(6):164:1–164:10, December 2011.

[138] Ahmed A. Shabana. *Theory of Vibration, Volume II: Discrete and Continuous Systems*. Springer–Verlag, New York, NY, 1990.

[139] Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph.*, 23(3):896–904, August 2004.

[140] Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. Computer puppetry: An importance-based approach. *ACM Trans. Graph.*, 20(2):67–94, April 2001.

[141] Patricio Simari, Evangelos Kalogerakis, and Karan Singh. Folding meshes: Hierarchical mesh segmentation based on planar symmetry. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 111–119, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[142] Wojciech Smigaj, S Arridge, T Betcke, Joel Phillips, and Martin Schweiger. Solving boundary integral problems with bem++. *ACM Transactions on Mathematical Software*, 2012.

[143] Breannan Smith, Danny M. Kaufman, Etienne Vouga, Rasmus Tamstorf,

and Eitan Grinspun. Reflections on Simultaneous Impact. *SIGGRAPH (ACM Transactions on Graphics)*, 31(4):106:1—-106:12, 2012.

[144] Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *Proceedings of Shape Modeling International*, pages 191–199. IEEE Computer Society Press, 2004.

[145] Donald E Spiel. Acoustical measurements of air bubbles bursting at a water surface: Bursting bubbles as helmholtz resonators. *Journal of Geophysical Research: Oceans (1978–2012)*, 97(C7):11443–11452, 1992.

[146] Kyle S. Spratt, Kevin M. Lee, Preston S. Wilson, and Mark S. Wochner. On the resonance frequency of an ideal arbitrarily-shaped bubble. *Proceedings of Meetings on Acoustics*, 20(1), 2015.

[147] Jos Stam. Stable fluids. In *Proc. ACM SIGGRAPH*, pages 121–128, 1999.

[148] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Trans. Graph.*, 32(4):102:1–102:10, July 2013.

[149] M. Strasberg. The pulsation frequency of nonspherical gas bubbles in liquids. *The Journal of the Acoustical Society of America*, 25(3):536–537, 1953.

[150] Shinjiro Sueda, Andrew Kaufman, and Dinesh K. Pai. Musculotendon simulation for hand animation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 27(3), 2008.

[151] Tapio Takala and James Hahn. Sound rendering. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, volume 26, pages 211–220, July 1992.

[152] Jie Tan, Yuting Gu, C. Karen Liu, and Greg Turk. Learning bicycle stunts. *ACM Trans. Graph.*, 33(4):50:1–50:12, 2014.

[153] Jie Tan, Yuting Gu, Greg Turk, and C. Karen Liu. Articulated swimming creatures. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 58:1–58:12. ACM, 2011.

[154] Diane Tang, J Thomas Ngo, and Joe Marks. N-body spacetime constraints. *The Journal of Visualization and Computer Animation*, 6(3 (July-Sept.)):143–154, 1995.

[155] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, pages 269–278, August 1988.

[156] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically Deformable Models. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, pages 205–214, July 1987.

[157] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulations. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 22(3):716–723, 2003.

[158] Nicolas Tsingos, Thomas Funkhouser, Addy Ngan, and Ingrid Carlbom. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 545–552, New York, NY, USA, 2001. ACM.

[159] Nicolas Tsingos, Emmanuel Gallo, and George Drettakis. Perceptual audio rendering of complex virtual environments. *ACM Trans. Graph.*, 23(3):249–258, August 2004.

[160] Christopher D. Twigg and Doug L. James. Many-worlds browsing for control of multibody dynamics. *ACM Transactions on Graphics*, 26(3):14:1–14:8, July 2007.

[161] Christopher D. Twigg and Doug L. James. Backward steps in rigid body simulation. *ACM Transactions on Graphics*, 27(3):25:1–25:10, August 2008.

[162] K. van den Doel and D. K. Pai. Synthesis of shape dependent sounds with physical modeling. In *Intl Conf. on Auditory Display*, Xerox PARC, Palo Alto, November 1996.

[163] Kees van den Doel. Physically based models for liquid sounds. *ACM Transactions on Applied Perception*, 2(4):534–546, October 2005.

[164] Kees van den Doel, Dave Knott, and Dinesh K. Pai. Interactive simulation of complex audiovisual scenes. *Presence: Teleoper. Virtual Environ.*, 13(1):99–111, 2004.

[165] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. FoleyAutomatic: Physically Based Sound Effects for Interactive Simulation and Animation.

In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 537–544, August 2001.

[166] Joshua D Wilson and Nicholas C Makris. Quantifying hurricane destructive power, wind speed, and air-sea material exchange with natural undersea sound. *Geophysical Research Letters*, 35(10), 2008.

[167] Andrew Witkin and Michael Kass. Spacetime constraints. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, pages 159–168, New York, NY, USA, 1988. ACM.

[168] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

[169] Yizhong Zhang, Chunji Yin, Changxi Zheng, and Kun Zhou. Computational hydrographic printing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2015)*, 34(4), August 2015.

[170] Changxi Zheng and Doug L. James. Harmonic fluids. *ACM Trans. Graph.*, 28(3):37:1–37:12, August 2009.

[171] Changxi Zheng and Doug L. James. Rigid-Body Fracture Sound with Precomputed Soundbanks. *ACM Transactions on Graphics*, 29(4):69:1–69:13, July 2010.

[172] Changxi Zheng and Doug L. James. Toward high-quality modal contact sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(4), August 2011.

[173] Wen Zheng, Jun-Hai Yong, and Jean-Claude Paul. Simulation of bubbles. *Graphical Models*, 71(6):229–239, 2009.